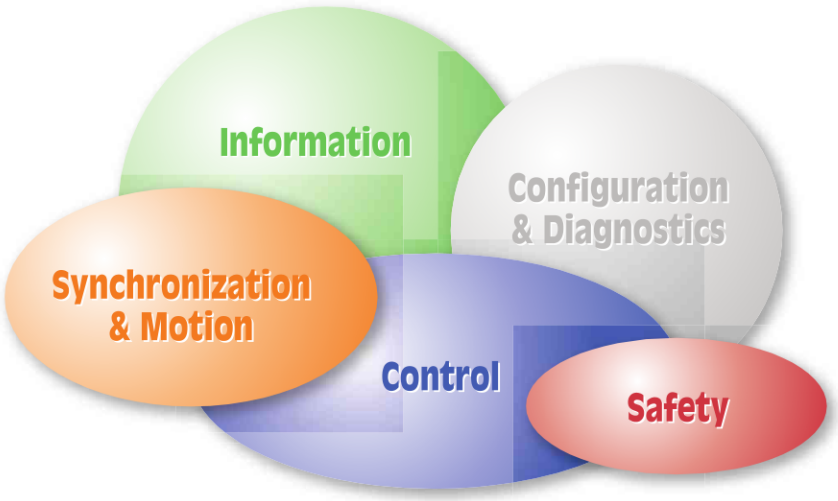
The Common Industrial Protocol (CIP™) and the Family of CIP Networks



The Common Industrial Protocol (CIP™) and the Family of CIP Networks

The Common Industrial Protocol (CIP™) and the Family of CIP Networks

Publication Number: PUB00123R0

Copyright © 2006 Open DeviceNet Vendor Association, Inc. (ODVA). All rights reserved. For permissions to reproduce excerpts of this material, with appropriate attribution to the author(s), please contact ODVA at:

ODVA
1099 Highland Drive, Suite A
Ann Arbor, MI 48108-5002 USA
TEL 1-734-975-8840
FAX 1-734-922-0027
EMAIL odva@odva.org
WEB www.odva.org

The right to make, use or sell product or system implementations described herein is granted only under separate license pursuant to a Terms of Usage Agreement or other agreement. Terms of Usage Agreements for individual CIP Networks are available, at standard charges, over the Internet at the following web sites:

www.odva.org - Terms of Usage Agreements for DeviceNet and EtherNet/IP along with general information on DeviceNet, CompoNet and EtherNet/IP networks and the association of ODVA

www.controlnet.org - Terms of Usage Agreement for ControlNet along with general information on ControlNet and ControlNet International.

Warranty Disclaimer Statement

Because CIP Networks my be applied in many diverse situations and in conjunction with products and systems from multiple vendors, the user and those responsible for specifying CIP Networks must determine for themselves its suitability for the intended use. The Specifications are provided to you on an AS IS basis without warranty. **NO WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE ARE BEING PROVIDED BY THE PUBLISHER, ODVA AND/OR CONTROLNET INTERNATIONAL.** In no event shall the Publisher, ODVA and/or ControlNet International, their officers, directors, members, agents, licensors or affiliates be liable to you or any customer for lost profits, development expenses or any other direct, indirect incidental, special or consequential damages.

ControlNet and ControlNet CONFORMANCE TESTED are trademarks of ControlNet International, Ltd.

CIP, DeviceNet and DeviceNet CONFORMANCE TESTED, EtherNet/IP CONFORMANCE TESTED, CIP Safety and CompoNet are trademarks of Open DeviceNet Vendor Association, Inc.

EtherNet/IP is a trademark of ControlNet International under license by Open DeviceNet Vendor Association, Inc.

All other trademarks referenced herein are property of their respective owners.

PUBLISHER:
Open DeviceNet Vendor Association, Inc. (ODVA)

AUTHOR:
Viktor Schiffer – Rockwell Automation

EDITORS:
David J. VanGompel – Member of Technical Review Board, ODVA

Raymond A. Romito – Senior Project Engineer, Rockwell Automation

Katherine Voss – Executive Director, ODVA

ABOUT THE AUTHOR

Viktor Schiffer has almost 20 years of experience in the field of manufacturing automation and industrial networks. Mr. Schiffer graduated from RWTH Aachen in Germany with a degree of Diplom-Ingenieur in Electrical Engineering and earned a Master of Science degree in Electronic Instrumentation from the University College of Wales in Swansea in the United Kingdom. He currently serves as Engineering Manager for Rockwell Automation in Germany where he is responsible for third party support of the open CIP Networking technologies DeviceNet, ControlNet and EtherNet/IP.

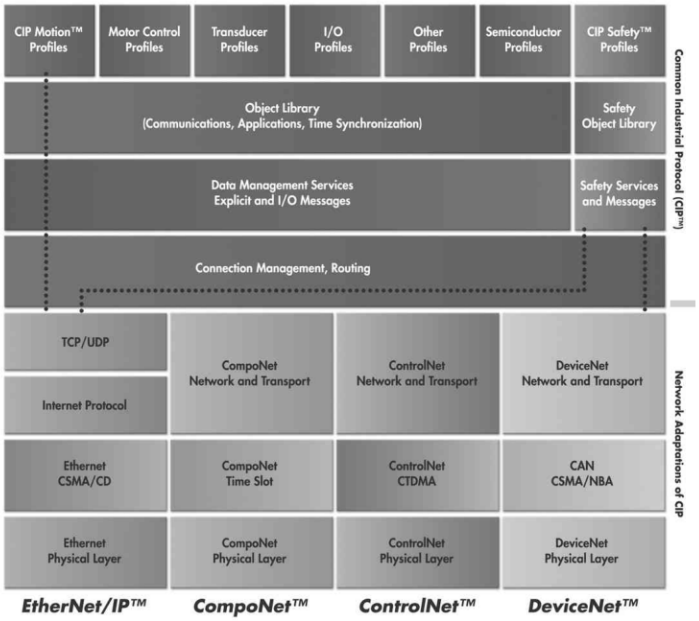
In addition, Mr. Schiffer is an active participant in ODVA where he contributes to the EtherNet/IP Implementor Workshops and PlugFests both in Europe and in North America. The author has been writing on the topic of industrial networks and has published a number of papers related to the CIP Networking technologies, including specifically on the use of Electronic Data Sheets and on real-time behavior of networks.

The Common Industrial Protocol (CIP™) and the Family of CIP Networks

Preface

Organization of the CIP Networks Specifications

Today, three networks - DeviceNet™, ControlNet™ and EtherNet/IP™ - use the Common Industrial Protocol (CIP) for the upper layers of their network protocol. For this reason, the associations that manage these networks - ODVA and ControlNet International - have mutually agreed to manage and distribute the specifications for CIP Networks in a common structure to help ensure consistency and accuracy in the management of these specifications. The following diagram illustrates the organization of the library of CIP Network specifications.



This common structure presents CIP in one volume with a separate volume for each network adaptation of CIP. The specifications for the CIP Networks are two-volume sets, paired as shown below.

The EtherNet/IP specification consists of:
Volume 1: Common Industrial Protocol (CIP™)
Volume 2: EtherNet/IP Adaptation of CIP

The DeviceNet specification consists of:
Volume 1: Common Industrial Protocol (CIP™)
Volume 3: DeviceNet Adaptation of CIP

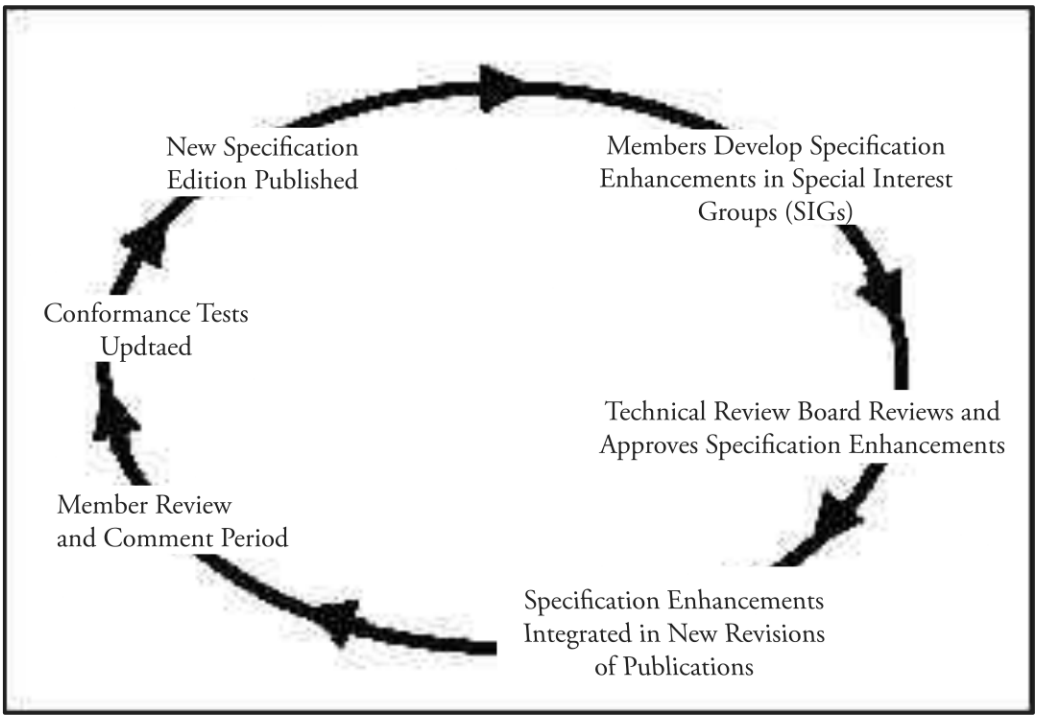
The ControlNet specification consists of:
Volume 1: Common Industrial Protocol (CIP™)
Volume 4: ControlNet Adaptation of CIP

The specifications for CIP Safety™ are distributed in a single volume:
Volume 5: CIP Safety

The forthcoming specification for CompoNet will consist of:
Volume 1: Common Industrial Protocol (CIP™)
Volume 6: CompoNet Adaptation of CIP

Specification Enhancement Process

The specifications for CIP Networks are continually being enhanced to meet the increasing needs of users for features and functionality. ODVA and ControlNet International have also agreed to operate using a common Specification Enhancement Process in order to ensure open and stable specifications for all CIP Networks. This process is on going throughout the year for each CIP Network Specification as shown in the *figure* below. New editions of each CIP Network specification are published on a periodic basis.



The Common Industrial Protocol (CIP™) and the Family of CIP Networks

1. Introduction

Traditionally, networks used in manufacturing enterprises have been optimized for performance in specific applications: most commonly, device, control, information and safety. While well suited to the functionality for which they were designed, these networks were not developed with a single, coherent enterprise architecture in mind. Since efficiency, reliability and, ultimately, profitability are generally dependent on having more than one of these capabilities, manufacturers have been forced to implement several different networks, none of which communicates innately with the other. As a result, most manufacturing enterprise network environments are characterized by numerous specialized—and generally incompatible—networks existing in one space.

Today, however, corporate expectations for the manufacturing automation network landscape have changed dramatically, thanks to the rapid and ubiquitous adoption of Internet technology. Companies of all sizes, all over the world, are trying to find the best ways to connect the entire enterprise. No longer is control of the manufacturing processes enough: the new manufacturing mandate is to enable users throughout the company to access manufacturing data from any location, at any time, and to integrate this data seamlessly with business information systems.

During recent years, a rapidly increasing number of users worldwide have looked to "open" systems as a way to connect their disparate enterprise processes. However, the great promise of open systems has often gone unfulfilled. The devices, programs and processes used at the various layers of the seven-layer Open System Interconnect (OSI) model have different options, capabilities and standards (or lack of). Integrating these networks requires extra resources and programming. Even then, gaps between the systems often cannot be fully and seamlessly bridged. Consequently, users compromise their investments and rarely achieve all of the productivity and quality benefits promised by open network technology.

Common application layers are the key to advanced communication and true network integration. The Common Industrial Protocol (CIP™), which is managed jointly by ODVA and ControlNet International, allows complete integration of control with information, multiple CIP Networks and Internet technologies. Built on a single, media-independent platform that provides seamless communication from the plant floor through the enterprise with a scalable and coherent architecture, CIP allows companies to integrate I/O control, device configuration and data collection across multiple networks. This ultimately helps minimize engineering and installation time and costs while maximizing ROI.

Introduced in 1994, DeviceNet™ is the first member of the CIP Family. DeviceNet is a CIP implementation using the popular Controller Area Network (CAN) data link layer. CAN in its typical form (ISO 11898[10]) defines only layers 1 and 2 of the OSI 7-layer model[13]. DeviceNet covers the remaining layers. Since DeviceNet has a low cost of implementation and is easy to use, many device manufacturers have built DeviceNet capable products. Several hundred of these manufacturers have organized in the Open DeviceNet Vendor Association (ODVA, http://www.odva.org).

ControlNet, introduced in 1997, implements the same basic protocol on new data link layers that allow for much higher speed (5 Mbps), strict determinism and repeatability while extending the range of the bus (several kilometers with repeaters) for more demanding applications. Vendors and users of ControlNet products are organized within ControlNet International (CI, http://www.controlnet.org) to promote the use of these products.

In 2000, ODVA and CI introduced another member of the CIP family: EtherNet/IP, where "IP" stands for "Industrial Protocol." In this network adaptation, CIP runs over TCP/IP and therefore can be deployed over any TCP/IP supported data link and physical layers, the most popular of which is IEEE 802.3[11], commonly known as Ethernet. The universal principles of CIP easily lend themselves to possible future implementations on new physical/data link layers, e.g., ATM, USB or FireWire. The overall relationship between the three implementations of CIP and the ISO/OSI 7-layer model is shown in *Figure 1*.

In 2004, ODVA introduced an extention to the CIP family of networks. CIP Safety provides functional safety for CIP Networks and provides users with fail-safe communication between devices, controllers and networks for safety applications.

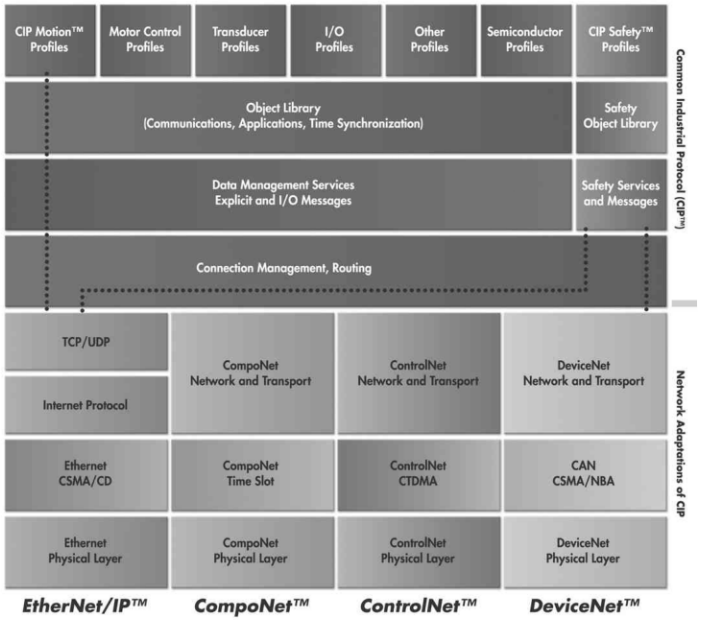


Figure 1 The CIP Family of Protocols

Two other significant additions to CIP are CIP Sync and CIP Motion. CIP Sync allows synchronization of applications in distributed systems through precision real-time clocks in all devices. These real-time clocks are kept in tight synchronization by background messages between clock masters and clock slaves using the new IEEE 1588:2002 standard[22]. This makes the CIP Sync technology ideally suited for motion control applications such as CIP Motion. A more detailed description of this CIP extension is given in *Section 5.1*. CIP Safety is a protocol extension that allows the transmission of safety relevant messages. Such messages are governed by additional timing and integrity mechanisms that are guaranteed to detect system flaws to a very high degree, as required by international standards such as IEC 61508[14]. If anything goes wrong, the system will be brought to a safe state, typically taking the machine to a standstill. A more detailed description of this CIP extension is given in *Section 5.2.* In both cases, ordinary devices can operate with CIP Sync or CIP Safety devices side by side in the same system. There is no need for strict segmentation into "Standard", "Sync" and "Safety" networks. It is even possible to have any combination of all three functions in one device.

2. Description of CIP

CIP is a very versatile protocol designed with the automation industry in mind. However, due to its open nature, it can be applied to many more areas. The CIP Networks Library contains several volumes:

- **Volume 1** details the common aspects of CIP that apply to all the network adaptations. This volume contains the Common Object Library and the Device Profile Library, along with a general description of the communications model, device configuration and CIP data management.
- **Volume 2** is the EtherNet/IP Adaptation of CIP, which describes how CIP is adapted to the Ethernet TCP/IP and UDP/IP transportation layers. It also contains any extensions to the material in Volume 1 that are necessary for EtherNet/IP.
- **Volume 3** is the DeviceNet Adaptation of CIP, which describes how CIP is adapted to the CAN data link layer. It also contains any extensions to the material in Volume 1 that are necessary for DeviceNet.
- **Volume 4** is the ControlNet Adaptation of CIP, which describes how CIP is adapted to the ControlNet data link layer. It contains a complete description of the ControlNet data link layer and any extensions to the material in Volume 1 that are necessary for ControlNet.
- **Volume 5** is CIP Safety. It contains the information necessary to implement the CIP Safety protocol on CIP Networks.

For brevity, this document will use the volume numbers above when referencing the different books in the CIP Networks Library.

Specifications for the CIP Networks referenced above, and other documents discussing CIP, are available from ODVA. It is beyond the scope of this book to fully describe each and every detail of CIP, but key features of the protocol will be discussed, including:

- Object Modeling
- Services
- Messaging Protocol
- Communication Objects
- Object Library
- Device profiles
- Configuration and electronic data sheets
- Bridging and Routing
- Data Management

A few terms used throughout this section are described here to make sure they are well understood:

Client: Within a client/server architecture, the client is the device that sends a request to a server. The client expects a response from the server.

Server: Within a client/server architecture, the server is the device that receives a request from a client. The server is expected to give a response to the client.

Producer: Within a producer/consumer architecture, the producing device places a message on the network for consumption by one or several consumers. Generally, the produced message is not directed to a specific consumer.

Consumer: Within a producer/consumer architecture, the consumer is one of potentially several consuming devices that picks up a message placed on the network by a producing device.

Producer/Consumer Model: CIP uses the producer/consumer model, as opposed to the traditional source/destination message addressing scheme *(see Figure 2)*. The producer/consumer model is inherently multicast. Nodes on the network determine if they should consume the data in a message based on the connection ID in the packet.

Explicit Message: Explicit messages contain addressing and service information that directs the receiving device to perform a certain service (action) on a specific part (e.g., an attribute) of a device.

Implicit (I/O) Message: Implicit messages do not carry address and/or service information; the consuming node(s) already know what to do with the data based on the connection ID that was assigned when the connection was established. Implicit messages are so named because the meaning of the data is "implied" by the connection ID.

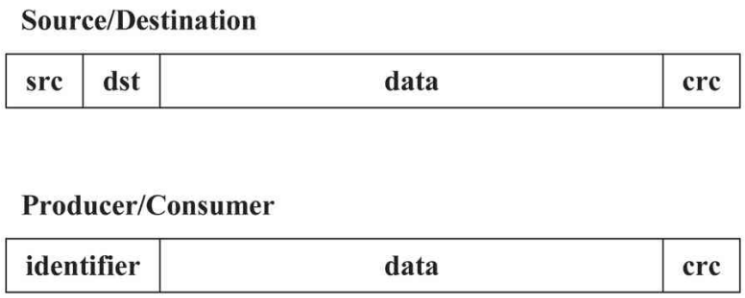


Figure 2 Source/Destination vs. Producer/Consumer Model

Let's now have a look at the individual elements of CIP:

2.1. Object Modeling

CIP uses abstract object modeling to describe:

- The suite of available communication services;
- The externally visible behavior of a CIP node;
- A common means by which information within CIP products is accessed and exchanged.

Every CIP node is modeled as a collection of objects. An object provides an abstract representation of a particular component within a product. Anything not described in object form is not visible through CIP. CIP objects are structured into classes, instances and attributes.

A **class** is a set of objects that all represent the same kind of system component. An object **instance** is the actual representation of a particular object within a class. Each instance of a class has the same **attributes**, but also has its own particular set of attribute values. As *Figure 3* illustrates, multiple object instances within a particular class can reside within a CIP node.

In addition to the instance attributes, an object class may also have class attributes. These are attributes that describe properties of the whole object class, e.g., how many instances of this particular object exist. Furthermore, both object instances and the class itself exhibit a certain behavior and allow certain services to be applied to the attributes, instances or to the whole class. All publicly defined objects that are implemented in a device must follow at least the mandatory requirements defined in the various CIP Networks specifications. Vendor-specific objects may also be defined with a set of instances, attributes and services according to the requirements of the vendor. However, they need to follow certain rules that are also set forth in the specifications.

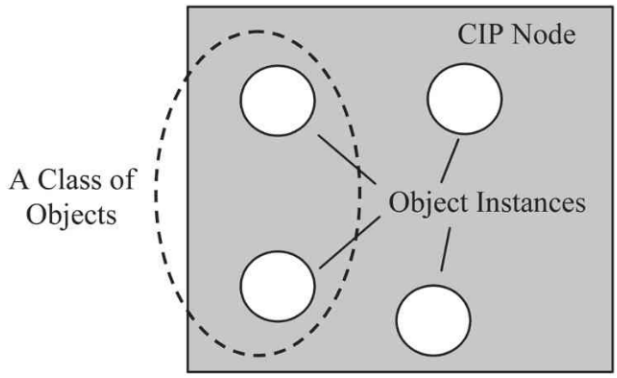


Figure 3 A Class of Objects

The objects and their components are addressed by a uniform addressing scheme consisting of:

Node Address: An integer identification value assigned to each node on a CIP Network. On DeviceNet and ControlNet, this is also called a MAC ID (Media Access Control Identifier) and is nothing more than the node number of the device. On EtherNet/IP, the node address is the IP address.

Class Identifier (Class ID): An integer identification value assigned to each object class accessible from the network.

Instance Identifier (Instance ID): An integer identification value assigned to an object instance that identifies it among all instances of the same class.

Attribute Identifier (Attribute ID): An integer identification value assigned to a class or instance attribute.

Service Code: An integer identification value which denotes an action request that can be directed at a particular object instance or object class *(see Section 2.2.)*.

Object class identifiers are divided into two types of open objects: publicly defined (ranging from 0x00 – 0x63 and 0x00F0 – 0x02FF) and vendor-specific objects (ranging from 0x64 – 0xC7 and 0x0300 – 0x04FF). All other class identifiers are reserved for future use. In some cases, e.g., within the assembly object class, instance identifiers are divided into two types of open instances: publicly defined (ranging from 0x01 – 0x63 and 0x0100 – 0x02FF) and vendor-specific (ranging from 0x64 – 0xC7 and 0x0300 – 0x04FF). All other instance identifiers are reserved for future use. Attribute identifiers are divided into two types of open attributes: publicly defined (ranging from 0x00 – 0x63) and vendor-specific (ranging from 0x64 – 0xC7). All other attribute identifiers are reserved for future use. While vendor-specific objects can be created with a great deal of flexibility, these objects must adhere to certain rules specified for CIP, e.g., they can use whatever instance and attribute IDs the developer wishes, but their class attributes must follow guidelines detailed in the CIP Volume section of each network specification.

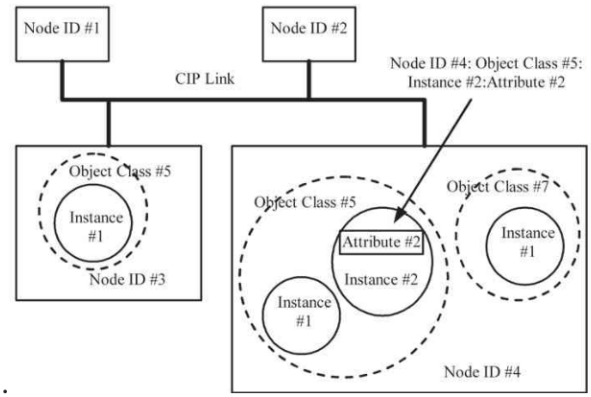


Figure 4 Object Addressing Example

Figure 4 shows an example of this object addressing scheme.

2.2. Services

Service codes are used to define the action that is requested to take place when an object or parts of an object are addressed through explicit messages using the addressing scheme described in *Section 2.1*. Apart from simple read and write functions, a set of CIP services has been defined. These CIP services are common in nature, meaning they may be used in all CIP Networks and they are useful for a variety of objects. Furthermore, there are object-specific service codes that may have a different meaning for the same code, depending on the class of object. Finally, defining vendor-specific services according to the requirements of the product developer is possible. While this provides a lot of flexibility, the disadvantage of vendor-specific services is that they may not be understood universally. Minimally, vendors provide a description of the public information that their customers will need access to in their literature.

2.3. Messaging Protocol

CIP is a connection-based protocol. A CIP connection provides a path between multiple application objects. When a connection is established, the transmissions associated with that connection are assigned a Connection ID, or CID *(see Figure 5)*. If the connection involves a bi-directional exchange, then two CID values are assigned.

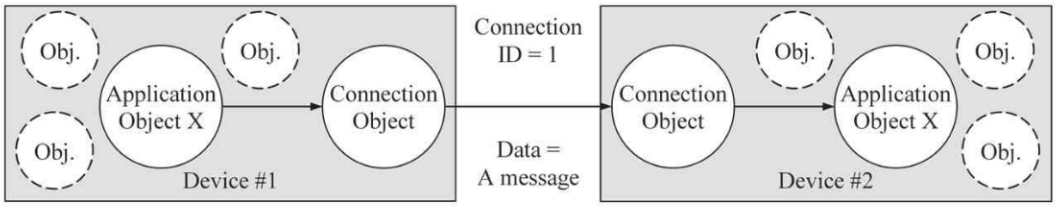


Figure 5 Connections and Connection IDs

The definition and format of the CID is network dependent. For example, the CID for CIP connections over DeviceNet is based on the CAN identifier field.

Since most messaging on a CIP Network is done through connections, a process has been defined to establish such connections between devices that are not yet connected. This is done through the Unconnected Message Manager (UCMM) function, which is responsible for processing unconnected explicit requests and responses.

Establishing a CIP connection is generally accomplished by sending a UCMM Forward_Open service request message. However, while this method is used on ControlNet and EtherNet/IP (all devices that support connected messaging support it), it is rarely used on DeviceNet, which typically uses the simplified methods described in *Sections 3.1.11. and 3.1.12*. DeviceNet Safety™ *(see Section 5.2.)*, on the other hand, fully utilizes this service.

A Forward_Open request contains all information required to create a connection between the originator and the target device and, if requested, a second connection between the target and the originator. In particular, the Forward_Open request contains information on the following:

- Time-out information for this connection;
- Network CID for the connection from the originator to the target;
- Network CID for the connection from the target to the originator;
- Information about the identity of the originator (vendor ID and serial number);
- Maximum data sizes of the messages on this connection;
- Trigger mechanisms, e.g. cyclic, change of state (COS);
- Connection path for the application object data in the node.

The connection path may also contain a routing segment that allows connections to exist across multiple CIP Networks. The Forward_Open request may also contain an electronic key of the target device (vendor ID, device type, product code and revision), as well as configuration information that will be forwarded to the configuration assembly of the target device.

Some networks, like ControlNet and EtherNet/IP, may also make use of unconnected explicit messaging. DeviceNet only uses unconnected messaging to establish connections.

All connections on a CIP Network can be categorized as I/O connections or explicit messaging connections.

- **I/O connections** provide dedicated, special-purpose communication paths between a producing application and one or more consuming applications. Application-specific I/O data move through these ports, a process that is often referred to as implicit messaging. These messages are typically multicast.
- **Explicit messaging connections** provide generic, multi-purpose communication paths between two devices. These connections often are referred to simply as messaging connections. Explicit messages provide typical request/response-oriented network communications. These messages are point-to-point.

The actual data transmitted in CIP I/O messages are the I/O data in an appropriate format; for example, the data may be prefixed by a sequence count value. This sequence count value can be used to distinguish old data from new, e.g., if a message has been re-sent as a heartbeat in a COS connection. The two states “run” and “idle” can be indicated with an I/O message either by prefixing a real-time header, as is the case for ControlNet and EtherNet/IP, or by sending I/O data (run) or no I/O data (idle), a process primarily used for DeviceNet. “Run” is the normal operative state of a device, while the reaction to receiving an “idle” event is vendor-specific and application-specific. Typically, this means bringing all outputs of the device to an “idle” (which usually means “off”) state, i.e., de-energized.

Explicit messaging requests, on the other hand, contain a service code with path information to the desired object (attribute) within the target device followed by data (if any). The associated responses repeat the service code followed by status fields followed by data (if any). DeviceNet uses a “condensed” format for explicit messages, while ControlNet and EtherNet/IP use the “full” format.

2.4. Communication Objects

CIP communication objects manage and provide the run-time exchange of messages. While these objects follow the overall principles and guidelines for CIP objects, communication objects are unique in that they are the focal points for all CIP communication. It therefore makes sense to look at them in more detail.

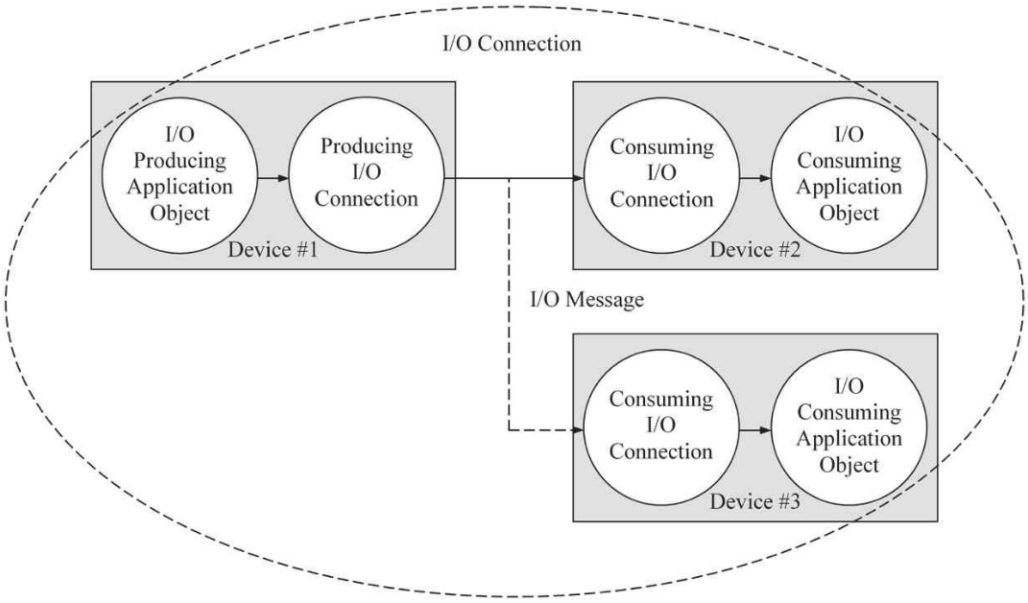


Figure 6 CIP Multicast I/O Connection

Each communication object contains a link producer part, a link consumer part or both. I/O connections may be either producing or consuming or producing and consuming, while explicit messaging connections are always producing and consuming.

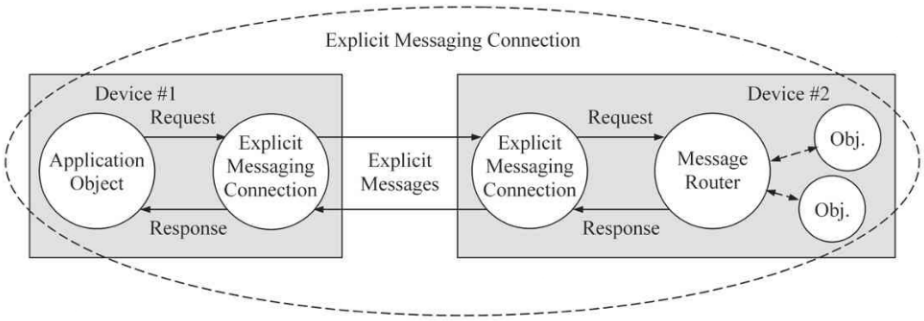


Figure 7 CIP Explicit Messaging Connection

Figure 6 and 7 show the typical connection arrangement for CIP I/O messaging and CIP explicit messaging. The attribute values in the connection objects define a set of attributes that describe vital parameters of this connection. Note that explicit messages are always directed to the message router object.

The attribute values of a connection object specify whether it is an I/O connection or an explicit messaging connection, the maximum size of the data to be exchanged across this connection and the source and destination of the data. Further attributes define the state and behavior of the connection. Particularly important behaviors include how messages are triggered (from the application, through change of state or change of data, through cyclic events or by network events) and the timing of the connections (time-out associated with this

connection and predefined action if a time-out occurs). CIP allows multiple connections to coexist in a device, although simple devices—e.g., simple DeviceNet slaves—typically will only have one or two live connections at any time.

2.5. Object Library

The CIP Family of Protocols contains a large collection of commonly defined objects. The overall set of object classes can be subdivided into three types:

• General-use;
• Application-specific;
• Network-specific.

Objects defined in Volume 1 of the CIP Networks Library are available for use on all network adaptations of CIP. Some of these objects may require specific changes or limitations when implemented on some of the network adaptations. These exceptions are noted in the network-specific volume. Therefore, to see a complete picture of a specific network implementation of an object, refer to Chapter 5 in both the Protocol Adaptation Volume and Volume 1.

The following are general use objects:

- Assembly
- Acknowledge Handler
- Connection
- Connection Configuration
- Connection Manager
- File
- Identity
- Message Router
- Parameter
- Parameter Group
- Port
- Register
- Selection

The following group of objects is application-specific:

- AC/DC Drive
- Analog Group
- Analog Input Group
- Analog Output Group
- Analog Input Point
- Analog Output Point
- Block Sequencer
- Command Block
- Control Supervisor
- Discrete Group
- Discrete Input Group
- Discrete Output Group
- Discrete Input Point
- Discrete Output Point
- Group
- Motor Data
- Overload
- Position Controller
- Position Controller Supervisor
- Position Sensor
- Presence Sensing
- S-Analog Actor
- S-Analog Sensor
- S-Device Supervisor
- S-Gas Calibration
- S-Partial Pressure
- S-Single Stage Controller
- Safety Supervisor
- Safety Validator
- Softstart Starter
- Trip Point

The last group of objects is network-specific:

- ControlNet
- ControlNet Keeper
- ControlNet Scheduling
- DeviceNet
- Ethernet Link
- TCP/IP Interface

The general-use objects can be found in many different devices, while the application-specific objects are typically found only in devices hosting such applications. New objects are added on an ongoing basis.

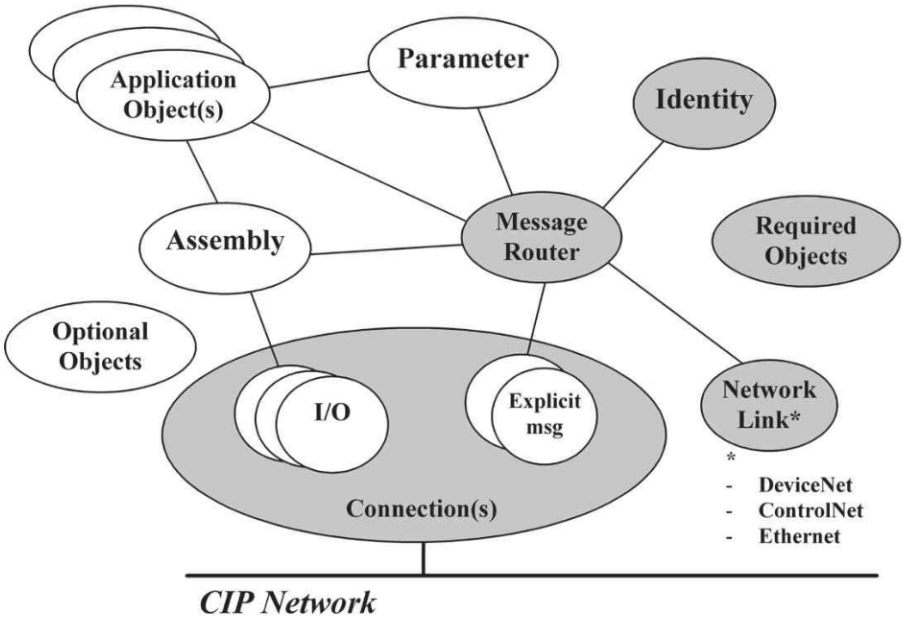


Figure 8 Typical Device Object Model

Although this looks like a large number of object types, typical devices implement only a subset of these objects. *Figure 8* shows the object model of such a typical device.

The objects required in a typical device are:

- Either a Connection Object or a Connection Manager Object
- An Identity Object
- One or several network-specific link objects (depends on network)
- A Message Router Object (at least its function)

Further objects are added according to the functionality of the device. This enables scalability for each implementation so that small devices, such as proximity sensors on DeviceNet, are not burdened with unnecessary overhead. Developers typically use publicly defined objects (see above list), but can also create their own objects in the vendor-specific areas, e.g., Class ID 100 – 199. However, they are strongly encouraged to work with the (Joint) Special Interest Groups (JSIGs/SIGs) of ODVA and ControlNet International to create common definitions for additional objects instead of inventing private ones.

Out of the general use objects, several will be described in more detail:

2.5.1. Identity Object (Class ID: 0x01)

The Identity Object is described in greater detail because, being a relatively simple object, it can serve to illustrate the general principles of CIP objects. In addition, every device must have an Identity Object.

The vast majority of devices support only one instance of the Identity Object. Thus, typically there are no requirements for any class attributes that describe additional class details, e.g., how many instances exist in the device. Only instance attributes are required in most cases. These are as follows:

Mandatory Attributes:

- Vendor ID
- Device Type
- Product Code
- Revision
- Status
- Serial Number
- Product Name

Optional Attributes:

- State
- Configuration Consistency Value
- Heartbeat Interval
- Languages Supported

Let us have a look at these attributes in more detail:

- The **Vendor ID** attribute identifies the vendor of every device. This UINT (Unsigned Integer) value for Data Type descriptions *(see Section 2.9.)* is assigned to a specific vendor by ODVA or ControlNet International. If a vendor intends to build products for more than one CIP Network, he will get the same Vendor ID for all networks.
- The **Device Type** specifies which profile has been used for this device. It must be one of the Device Types described in CIP or a vendor-specific type *(see Section 2.6.)*.
- The **Product Code** is a UINT number defined by the vendor of the device. This code is used to distinguish multiple products of the same Device Type from the same vendor.
- The **Revision** is split into two USINT (Unsigned Short Integer) values specifying a Major Revision and a Minor Revision. Any device change(s) that result in modifying the behavior of the device on the network must be reflected in a change to the Minor Revision at minimum. Any change(s) in the device requiring a revised Electronic Data Sheet (EDS) *(see Section 2.7.)* must be reflected in a change to the Major Revision. Vendor ID, Device Type, Product Code and Major Revision provide an unambiguous identification of an EDS for this device.
- The **Status** attribute provides information on the status of the device, e.g., whether it is owned (controlled by another device) or configured (to something different from the out-of-the-box default), and whether any major or minor faults have occurred.
- The **Serial Number** is used to uniquely identify individual devices in conjunction with the Vendor ID, i.e., no two CIP devices from one vendor may carry the same Serial Number. The 32 bits of the Serial Number allow ample space for a subdivision into number ranges that can be used by different divisions of larger companies.
- The **Product Name** attribute allows the vendor to give a meaningful ASCII name string (up to 32 characters) to the device.
- The **State** attribute describes the state of a device in a single UINT value. It is thus less detailed than the Status attribute.
- The **Configuration Consistency Value** allows a distinction between a device that has been configured and one that has not, or between different configurations in a single device. This helps avoid unnecessary configuration downloads.
- The **Heartbeat** Interval enables the Device Heartbeat Message. The maximum time between two heartbeats can be set between one and 255 seconds.

The services the class and instance attributes support are either Get_Attribute_Single (typically implemented in DeviceNet devices) or Get_Attributes_All (typically implemented in ControlNet and EtherNet/IP devices). None of the attributes can be set, except for the Heartbeat Interval, if implemented. The only other service that typically is supported by the Identity Object is the Reset service.

The behavior of the Identity Object is described through a state transition diagram.

2.5.2. Parameter Object (Class ID: 0x0F)

This object is described in some detail since it is referred to in *Section 2.7.*, Configuration and Electronic Data Sheets. When used, the Parameter Object comes in two "flavors:" a complete object and an abbreviated version (Parameter Object Stub). This abbreviated version is used primarily by DeviceNet devices that have only small amounts of memory available. The Parameter Object Stub, in conjunction with the Electronic Data Sheet, has more or less the same functionality as the full object *(see Section 2.7.)*.

The purpose of the Parameter Object is to provide a general means of allowing access to many attributes of the various objects in the device without requiring a simple tool (such as a handheld terminal) to have any knowledge about specific objects in the device.

The class attributes of the Parameter Object contain information about how many instances exist in this device and a Class Descriptor, indicating, among other properties, whether a full or a stub version is supported. In addition, they tell whether a Configuration Assembly is used and what language is used in the Parameter Object.

The first six Instance Attributes are required for the Object Stub. These are:

Parameter Value	This is the actual parameter.
Link Path Size **Link Path**	These two attributes describes the application object/ instance/attribute from which the parameter value was retrieved..
Descriptor	This describes parameter properties, e.g. read-only, monitor parameter etc.
Data Type	This describes the data type (size, range, etc) using a standard mechanism defined by CIP.
Data Size	Data size in bytes.

These six attributes allow access, interpretation and modification of the parameter value, but the remaining attributes make it much easier to understand the meaning of the parameter:

- The next three attributes provide ASCII strings with the name of the parameter, its engineering units and an associated help text;
- Another three attributes contain the minimum, maximum and default values of the parameter;
- Four more attributes can link the scaling of the parameter value so that the parameter can be displayed in a more meaningful way, e.g., raw value in multiples of 10 mA, scaled value displayed in Amps;
- Another four attributes can link the scaling values to other parameters. This feature allows variable scaling of parameters, e.g., percentage scaling to a full range value that is set by another parameter;

- Attribute #21 defines how many decimal places are to be displayed if the parameter value is scaled;
- Finally, the last three attributes are an international language version of the parameter name, its engineering units and the associated help text.

2.5.3. Assembly Object (Class ID: 0x04)

Assembly Objects provide the option of mapping data from attributes of different instances of various classes into one single attribute (#3): an Assembly Object. This mapping is generally used for I/O Messages to maximize the efficiency of the control data exchange on the network. Assembly mapping makes the I/O data available in one block; thus, there are fewer Connection Object instances and fewer transmissions on the network. The process data are normally combined from different application objects. An Assembly Object also can be used to configure a device with a single data block, alleviating the need to set individual parameters.

CIP makes a distinction between Input and Output Assemblies. "Input" and "Output" in this context are viewed from the perspective of the controlling element (e.g., a PLC). An Input Assembly in a device collects data from the input application (e.g., field wiring terminal, proximity sensor, etc.) and produces it on the network, where it is consumed by the controlling device and/or operator interface. An Output Assembly in a device consumes data that the controlling element sends to the network and writes that data to the output application (e.g., field wiring terminals, motor speed control, etc). This data mapping is very flexible; even mapping of individual bits is permitted. Assemblies also can be used to transmit a complete set of configurable parameters instead of accessing them individually. These Assemblies are called Configuration Assemblies.

Figure 9 shows an example of Assembly mapping. The data from application objects 100 and 101 are mapped in two instances of the Assembly Object. Instance 1 is set up as an Input Assembly for the input data, and instance 2 as an Output Assembly for output data. The data block is always accessed via attribute 3 of the relevant Assembly instance. Attributes 1 and 2 contain mapping information.

I/O Assembly mapping is specified for certain Device Profiles (e.g., Motor Starters) by ODVA. Device developers then can choose which Assemblies they support in their products. If none of the publicly defined Assemblies fully represents the functionality of the product, a device vendor may implement additional vendor-specific Assemblies (Instance IDs 100 – 199).

CIP defines static and dynamic Assembly Objects. Whereas mapping for static Assemblies is permanently programmed in the device (ROM), it can be modified and extended for dynamic mapping (RAM). Most simple CIP devices support only static Assembly Objects. Dynamic Assembly Objects tend to be used in more complex devices.

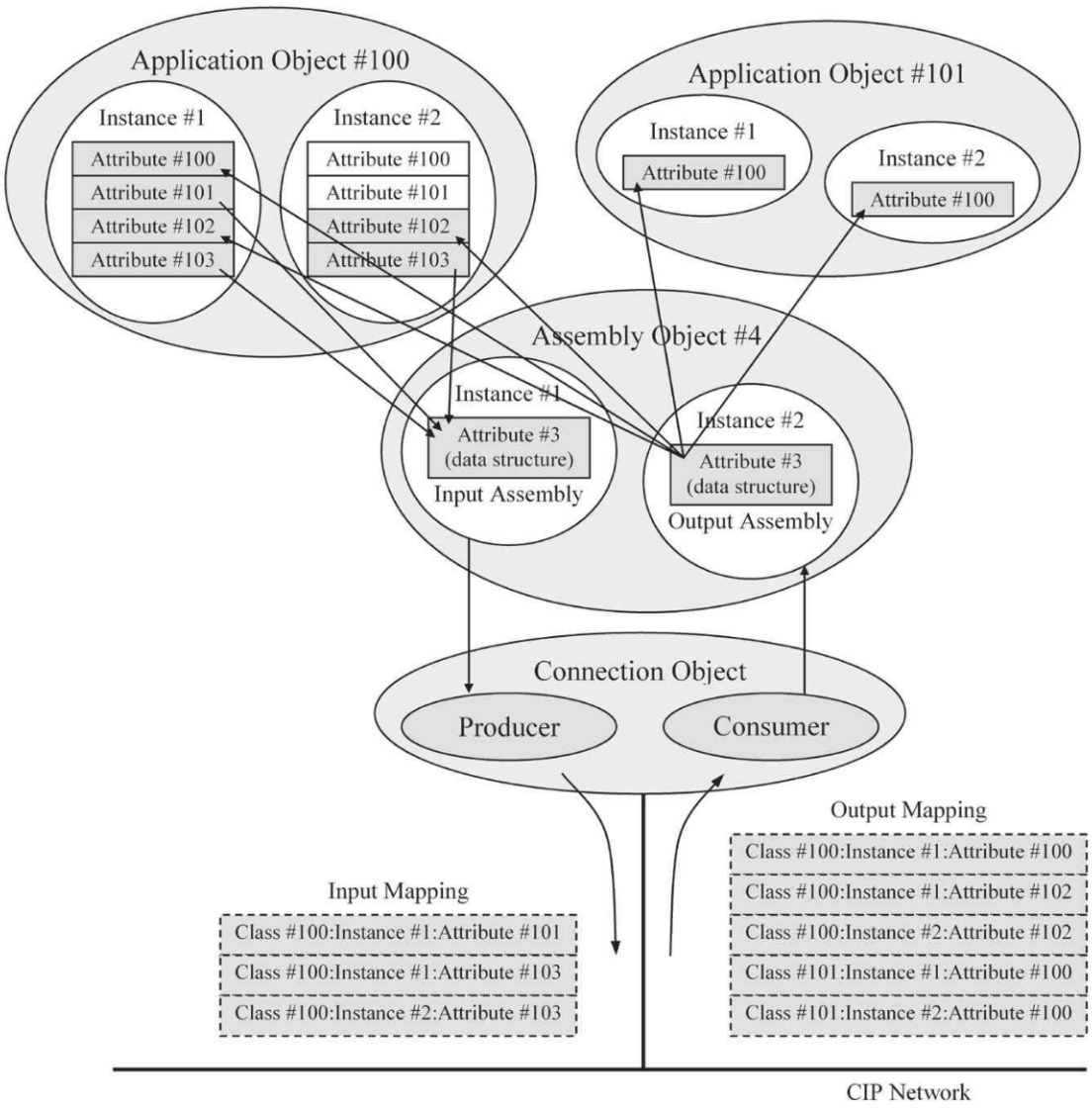


Figure 9 Example of an Assembly Mapping in a Typical I/O Device

2.6. Device Profiles

It would be possible to design products using only the definitions of communication networks and objects, but this could easily result in similar products having quite different data structures and behavior. To overcome this situation and to make the application of CIP devices much easier, devices of similar functionality have been grouped into Device Types with associated profiles. Such a CIP profile contains the full description of the object structure and behavior. The following Device Types and associated profiles are defined in Volume 1[1] (profile numbers are bracketed):

- AC Drives Device (0x02)
- Communications Adapter (0x0C)
- Contactor (0x15)
- ControlNet Physical Layer Component (0x32)
- ControlNet Programmable Logic Controller (0x0E)
- DC Drives (0x13)
- DC Power Generator (0x1F)
- Encoder (0x22)
- Fluid Flow Controller (0x24)
- General Purpose Discrete I/O (0x07)
- Generic Device (0x00)
- Human Machine Interface (0x18)
- Inductive Proximity Switch (0x05)
- Limit Switch (0x04)
- Mass Flow Controller (0x1A)
- Motor Overload Device (0x03)
- Motor Starter (0x16)
- Photoelectric Sensor (0x06)
- Pneumatic Valve (0x1B)
- Position Controller (0x10)
- Process Control Valve (0x1D)
- Residual Gas Analyzer (0x1E)
- Resolver (0x09)
- RF Power Generator (0x20)
- Safety Discrete I/O (0x23)
- Softstart Starter (0x17)
- Turbomolecular Vacuum Pump (0x21)
- Vacuum/Pressure Gauge (0x1C)

Device developers must use a profile. Any device that does not fall into the scope of one of the specialized profiles must use the Generic Device profile or a vendor-specific profile. What profile is used and which parts of it are implemented must be described in the user's device documentation.

Every profile consists of a set of objects—some required, some optional—and a behavior associated with that particular type of device. Most profiles also define one or several I/O data formats (Assemblies) that define the meaning of the individual bits and bytes of the I/O data. In addition to the publicly-defined object set and I/O data Assemblies, vendors can add objects and Assemblies of their own if their devices provide additional functionality. In addition, vendors can create profiles within the vendor-specific profile range. They are then free to define whatever behavior and objects are required for their device as long as they adhere to some general rules for profiles. Whenever additional functionality is used by multiple vendors, ODVA and ControlNet International encourage coordinating these new features through discussion in the Joint Special Interest Groups (JSIGs), which can then create new profiles and additions to existing profiles for everybody's use and for the benefit of the device users.

All open (ODVA/CI defined) profiles carry numbers in the 0x00 through 0x63 or 0x0100 through 0x02FF ranges, while vendor-specific profiles carry numbers in the 0x64 through 0xC7 or 0x0300 through 0x02FF ranges. All other profile numbers are reserved by CIP.

2.7. Configuration and Electronic Data Sheets

CIP provides several options for configuring devices:

• A printed data sheet;
• Parameter Objects and Parameter Object Stubs;
• An Electronic Data Sheet (EDS);
• A combination of an EDS and Parameter Object Stubs;
• A Configuration Assembly combined with any of the above methods.

When using configuration information collected on a printed data sheet, configuration tools can only provide prompts for service, class, instance and attribute data and relay this information to a device. While this procedure can do the job, it is the least desirable solution since it does not determine the context, content or format of the data

Parameter Objects, on the other hand, provide a full description of all configurable data for a device. Since the device itself provides all the necessary information, a configuration tool can gain access to all parameters and maintain a user-friendly interface. However, this method imposes a burden on a device with full parameter information that may be excessive for a small device, e.g., a simple DeviceNet slave. Therefore, an abbreviated version of the Parameter Object, called a Parameter Object Stub, may be used *(see Section 2.5.2.)*. This option still allows access to the parameter data, but it does not describe any meaning to the data. Parameter Stubs in conjunction with a printed data sheet are usable, but certainly not optimal. On the other hand, an EDS supplies all of the information that a full Parameter Object contains, in addition to I/O Connection information. The EDS thus provides the full functionality and ease of use of the Parameter Object without imposing an excessive burden on the individual device. In addition, an EDS provides a means for tools to perform offline configuration and to download configuration data to the device at a later time.

An EDS is a simple ASCII text file that can be generated on any ASCII editor. Since the CIP Specification lays down a set of rules for the overall design and syntax of an EDS which makes configuration of devices much easier. Specialized EDS editing tools, such as ODVA's EZ-EDS, can simplify the creation of EDS files. The main purpose of the EDS is to give information on several aspects of the device's capabilities, the most important ones being the I/O Connections it supports and what parameters for display or configuration exist within the device. It is highly recommended that an EDS describe all supported I/O Connections, as this makes the application of a device much easier. When it comes to parameters, it is up to the developer to decide which items to make accessible to the user.

Let's look at some details of the EDS. First, an EDS is structured into sections, each of which starts with a section name in square brackets []. The first two sections are mandatory for all EDSs.

- [File]: Describes the contents and revision of the file.
- [Device]: Is equivalent to the Identity Object information and is used to match an EDS to a device.
- [Device Classification]: Describes what network the device can be connected to. This section is optional for DeviceNet, required for ControlNet and EtherNet/IP.
- [IO_Info]: Describes connection methods & I/O sizes. Required for DeviceNet only.
- [Variant_IO_Info]: Describes multiple IO_Info data sets. Required for DeviceNet only.
- [ParamClass]: Describes class-level attributes of the Parameter Object.
- [Params]: Identifies all configuration parameters in the device, follows the Parameter Object definition. Further details below.
- [EnumPar]: Enumeration list of parameter choices to present to the user. This is an old method specified for DeviceNet only.
- [Assembly]: Describes the structure of data items.
- [Groups]: Identifies all parameter groups in the device and lists group name and Parameter Object instance numbers.
- [Connection Manager]: Describes connections supported by the device. Typically used in ControlNet and EtherNet/IP.
- [Port]: Describes the various network ports a device may have.
- [Modular]: Describes modular structures inside a device.
- [Capacity]: Specifies the communication capacity of EtherNet/IP and ControlNet devices.

Very detailed device descriptions can be made witnin these sections, although only a few of these details are described here. Further reading is available in endnotes[23-24].

A tool with a collection of EDSs will first use the [Device] section to try to match an EDS with each device it finds on a network. Once this is done and a particular device is chosen, the tool can then display device properties and parameters and allow their modification (if necessary). A tool may also display what I/O Connections a device may allow and which of these are already in use. EDS-based tools are mainly used for slave or adapter devices, as scanner devices typically are too complex to be configured through EDSs. For those devices, the EDS is used primarily to identify the device, then guide the tool to call a matching configuration applet.

A particular strength of the EDS approach lies in the methodology of parameter configuration. A configuration tool typically takes all of the information supplied by the Parameter Object and an EDS and displays it in a user-friendly manner. In many cases, this enables the user to configure a device without needing a detailed manual, as the tool presentation of the parameter information, together with help texts, enables decisions making for a complete device configuration (provided, of course, the developer has supplied all required information).

A complete description of what can be done with EDSs goes well beyond the scope of this book. Available materials on this topic provide greater detail[23-24].

2.8. Bridging and Routing

CIP defines mechanisms that allow the transmission of messages across multiple networks, provided the bridging devices (routers) between the various networks are equipped with a suitable set of capabilities (e.g., objects and support of services). If this is the case, the message will be forwarded from router to router until it has reached its destination node. Here is how it works:

For Unconnected Explicit Messaging, the actual Explicit Message to be executed on the target device is "wrapped up" using another type of Explicit Message service, the so-called Unconnected_Send message. The Unconnected_Send message (Service Code 0x52 of the Connection Manager Object) contains complete information on the transport mechanism, in particular, time-outs (they may be different while the message is still en route) and path information. The first router device that receives an Unconnected_Send message will take its contents and forward it to the next network as specified within the path section of the message. Before the message is actually sent, the "used" part of the path is removed, but is remembered by the intermediate router device for the return of any response. This process is executed for every "hop" until the final destination network is reached. The number of hops is theoretically limited by the message length.

Once the Unconnected_Send message has arrived at the target network, the "inner" Explicit Message is sent to the target device, which executes the requested service and generates a response. The response is then transported back through all the routers it traversed during its forward journey until it finally reaches the originating node. It is important to note in this context that the transport mechanism may have been successful in forwarding the message and returning the response, but the response still could contain an indication that the desired service could not be performed successfully in the target network/device. Through this mechanism, the router devices do not need to know anything about the message paths ahead of time. Thus, no programming of any of the router devices is required. This is often referred to as "seamless routing."

When a connection (I/O or Explicit) is set up using the Forward_Open service *(see Section 3.2.10.)*, it may go to a target device on another network. To enable the appropriate setup process, the Forward_Open message may contain a field with path information describing a route to the target device. This is very similar to the Unconnected_Send service described above. The routing information is then used to create routed connections within the routing devices between the originator and the target of the message. Once set up, these connections automatically forward any incoming messages for this connection to the outgoing port en route to the target device. Again, this is repeated until the message has reached its target node. As with routed Unconnected Explicit Messages, the number of hops is generally limited only by the capabilities of the devices involved. In contrast to routed Unconnected Messages, routed Connected Messages do not carry path information. Since Connected Messages always use the same path for any given connection, the path information that was given to the routing devices during connection setup is held there as long as the connection exists. Again, the routing devices do not have to be preprogrammed; they are self-configured during the connection establishment process.

2.9. Data Management

The Data Management part of the CIP Specification describes addressing models for CIP entities and the data structures of the entities themselves.

Entity addressing is done by Segments, which allows flexible usage so that many different types of addressing methods can be accommodated. The first byte of a CIP Segment allows a distinction between a Segment Address (0x00 – 0x9F) and a Data Type description (0xA0 – 0xDF). Two uses of this addressing scheme (Logical Segments and Data Types) are looked at in more detail below.

2.9.1. Logical Segments

Logical Segments (first byte = 0x20 – 0x3F) are addressing Segments that can be used to address objects and their attributes within a device. They are typically structured as follows: [Class ID] [Instance ID] [Attribute ID, if required].

Each element of this structure allows various formats (1 byte, 2 byte and 4 byte). *Figure 10* shows a typical example of this addressing method.

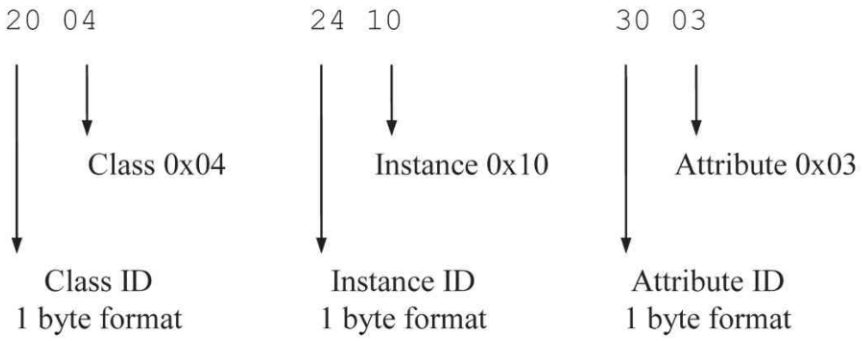


Figure 10 Logical Segment Encoding Example

This type of addressing is commonly used to point to Assemblies, Parameters and other addressable attributes within a device. It is used extensively in EDSs, but also within Unconnected Messages, to name just a few application areas.

2.9.2. Data Types

Data Types (first byte = 0xA0 – 0xDF) can be either structured (first byte 0xA0 – 0xA3) or elementary Data Types (first and only byte 0xC1 – 0xDE). Structured Data Types can be arrays of elementary Data Types or any assembly of arrays or elementary Data Types. Of particular importance in the context of this book are elementary Data Types, which are used within EDSs to specify the Data Types of parameters and other entities.

Here is a list of commonly used Data Types:

- 1-bit (encoded into 1-byte):
  - o Boolean, BOOL, Type Code 0xC1
- 1-byte:
  - o Bit string, 8 bits, BYTE, Type Code 0xD1
  - o Unsigned 8-bit integer, USINT, Type Code 0xC6
  - o Signed 8-bit integer, SINT, Type Code 0xC2
- 2-byte:
  - o Bit string, 16-bits, WORD, Type Code 0xD2
  - o Unsigned 16-bit integer, UINT, Type Code 0xC7
  - o Signed 16-bit integer, INT, Type Code 0xC3
- 4-byte:
  - o Bit string, 32-bits, DWORD, Type Code 0xD3
  - o Unsigned 32-bit integer, UDINT, Type Code 0xC8
  - o Signed 32-bit integer, DINT, Type Code 0xC4

The Data Types in CIP follow the requirements of IEC 61131-3[8].

2.9.3. Maintenance and Further Development of the Specifications

Both ODVA and ControlNet International have a set of working groups that maintain the specifications and create protocol extensions, e.g., new profiles or functional enhancements such as CIP Sync and CIP Safety. These groups are called Special Interest Groups (SIGs) for DeviceNet and ControlNet, and Joint Special Interest Groups (JSIGs) for EtherNet/IP, since the EtherNet/IP technology is jointly administered by both ODVA and ControlNet International members.

The results of these SIGs and JSIGs are written up as DSEs (DeviceNet Specification Enhancements), CSEs (ControlNet Specification Enhancements), ESEs (EtherNet/IP Specification Enhancements), SSEs (Safety Specification Enhancements) or CIPSEs (CIP Specification Enhancements), presented to the Technical Review Board (TRB) for approval and then incorporated into the specifications. Only ODVA or ControlNet International members can work within the SIGs and JSIGs, and participants have the advantage of advance knowledge of technical changes. Participation in one or several SIGs is, therefore, highly recommended.

3. Network Adaptations of CIP

Three public derivatives of CIP currently exist. These three derivatives are based on different data link layers and transport mechanisms, but they maintain the principles of CIP.

3.1. DeviceNet

DeviceNet was the first public implementation of CIP. As mentioned in *Section 1.*, DeviceNet is based on the Controller Area Network (CAN). DeviceNet uses a subset of the CAN protocol (11-bit identifier only, no remote frames). The DeviceNet adaptation of CIP accommodates certain limitations of the CAN protocol (which permits a maximum 8 bytes of payload) and allows the use of simple devices with only minimal processing power. For a more detailed description of the CAN protocol and some of its applications, see endnote[9].

Figure 11 shows the relationship between CIP, DeviceNet and the ISO/OSI layer model.

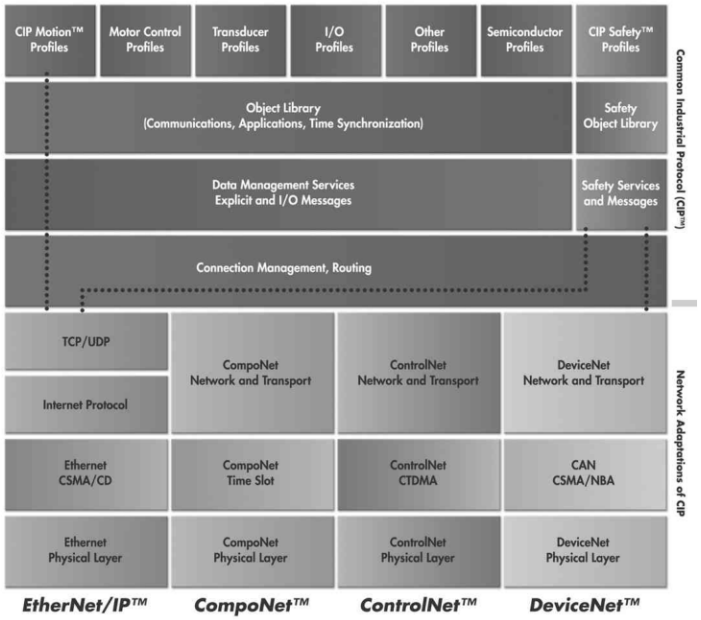


Figure 11 Relationship Between CIP and DeviceNet

3.1.1. Physical Layer and Relationship to CAN

The physical layer of DeviceNet is an extension of the ISO 11898 standard[10]. This extension defines the following additional details:

- Improved transceiver characteristics that allow up to 64 nodes per network;
- Additional circuitry for overvoltage and mis-wiring protection;
- Several types of cables for a variety of applications;
- Several types of connectors for open (IP20) and enclosed (IP65/67) devices.

These extensions result in a communication system with the following physical layer characteristics:

- Trunkline/dropline configuration;
- Support for up to 64 nodes;
- Node insertion or removal with the network on;
- Simultaneous support for both network-powered (sensors) and separately-powered (actuators) devices;

- Use of sealed or open-style connectors;
- Protection from wiring errors;
- Selectable data rates of 125 kBaud, 250 kBaud and 500 kBaud;
- Adjustable power configuration to meet individual application needs;
- High current capability (up to 16 Amps per supply);
- Operation with off-the-shelf power supplies;
- Power taps that allow the connection of several power supplies from multiple vendors that comply with DeviceNet standards;
- Built-in overload protection;
- Power available along the bus: both signal and power lines contained in the cable;
- Several cables have different current capacities suitable for different applications.

The cables described in The CIP Networks Library were designed specifically to meet minimum propagation speed requirements to ensure that they can be used up to the maximum system length. Using the specified cables, in conjunction with suitable transceiver circuits and proper termination resistors (121 Ω), results in overall systems as specified in *Figure 12*.

	Trunk Distance			Drop Length	
Data Rate	Thick Cable	Thin Cable	Flat Cable	Maximum	Cumulative
125 kBaud	500 meters	100 meters	420 meters	6 meters	156 meter
250 kBaud	250 meters		200 meters		78 meters
500 kBaud	100 meters		75 meters		39 meters

Figure 12 Data Rate vs. Trunk and Drop Length

ODVA has issued a guideline[6] that gives complete details how to build the physical layer of a DeviceNet Network.

Developers of DeviceNet devices can create DeviceNet circuits with or without physical layer isolation (both versions are fully specified). Furthermore, a device may take some or all of its power from the bus, thus avoiding extra power lines for devices that can live on the power supplied through the DeviceNet cable.

All DeviceNet devices must be equipped with one of the connectors described in Volume 3. Hard wiring of a device is allowed, provided the node is removable without severing the trunk.

3.1.2. Protocol Adaptations

On the protocol side, there are basically two adaptations of CIP (apart from the addition of the DeviceNet Object) that have been made to better accommodate it to the CAN data frame:

- Limitation to short messages (8 bytes or less) where possible, with the introduction of fragmentation for longer messages;
- Introduction of the Master/Slave communication profile minimizes connection establishment management *(see Section 3.1.12.)*.

These two features have been created to allow the use of simple and thus inexpensive microcontrollers. This is particularly important for small, cost-sensitive devices like photoelectric or proximity sensors. As a result of this specialization, the DeviceNet protocol in its simplest form has been implemented in 8-bit microprocessors with as little as 4 Kbytes of code memory and 175 bytes of RAM.

Message fragmentation comes in two varieties:

- For **I/O Messages**, which typically are sent with a fixed length, the use of fragmentation is defined by the maximum length of the data to be transmitted through a connection. Any connection that has more than 8 bytes to transmit always uses the fragmentation protocol, even if the actual data to be transmitted is 8 bytes or less, e.g., an "Idle" Message.
- For **Explicit Messaging**, the use of the fragmentation protocol is indicated with every message, since the actual message will vary in length. The actual fragmentation protocol is contained in one extra byte within the message that indicates whether the fragment is a start fragment, a middle fragment or an end fragment. A modulo 64 rolling fragment counter allows very long fragmented messages, and is limited in theory only by the maximum Produced or Consumed Connection sizes (65,535 bytes). In reality, the capabilities of the devices limit the message sizes.

3.1.3. Indicators and Switches

Indicators and switches are optional on DeviceNet. However, certain DeviceNet users not only require indicators and switches, they also specify what type to use. Many factors must be considered before implementing these devices, including packaging, accessibility and customer expectations.

Indicators allow the user to determine the state of the device and its network connection(s). Since indicators can be very useful when troubleshooting the operation of a device, manufacturers are advised to incorporate some of the indicators described in the DeviceNet specification. While devices may incorporate additional indicators with behavior not described in the specification, any indicators described in the specification must also follow their specified behavior.

Similarly, devices may be built with or without switches or other directly accessible means for configuration of MAC ID and baud rate. If these switches are used, certain rules apply to how these values are used at power-up and during the operation of the device.

3.1.4. Additional Objects

The DeviceNet Specification defines one additional object, the DeviceNet Object.

3.1.4.1. DeviceNet Object (Class ID = 0x03)

A DeviceNet Object is required for every DeviceNet port of the device. The instance attributes of this object contain information on how this device uses the DeviceNet port, including the MAC ID of the device and the (expected) baud rate of the DeviceNet Network the device is attached to. Both attributes are always expected to be non-volatile, i.e., after a power interruption, the device is expected to try to go online again with the same values that were stored in these attributes before the power interruption. Devices that set these values through switches typically override any stored values at power-up.

3.1.5. Network Access

DeviceNet uses the network access mechanisms described in the CAN specification, i.e., bitwise arbitration through the CAN Identifier for every frame to be sent. This requires a system design that does not allow multiple uses of any of these identifiers. Since the node number of every device is coded into the CAN Identifier *(see Section 3.1.10)*, it is generally sufficient to make sure that none of the node numbers exists more than once on any given network. This is guaranteed through the Network Access algorithm *(see Section 3.1.6).*

3.1.6. Going Online

Any device that wants to communicate on DeviceNet must go through a Network Access algorithm before any communication is allowed. The main purpose of this process is to avoid duplicate Node IDs on the same network. Every device that is ready to go online sends a Duplicate MAC ID Check Message containing its Port Number, Vendor ID and Serial Number. If another device is already online with this MAC ID or is in the process of going online, it responds with a Duplicate MAC ID Response Message that directs the checking device to go offline and not communicate any further.

If two or more devices with the same MAC ID happen to transmit the Duplicate MAC ID Check Message at exactly the same time, all of them will win arbitration at the same time and will proceed with their message. However, since this message has different values (the Vendor ID and serial number) in the data field, the nodes will detect bit errors and will transmit error frames that cause all nodes to discard the frame. This reaction triggers a re-transmission of the message by the sending node. While this action may eventually result in a Bus-Off condition for the devices involved, a situation with duplicate Node IDs is safely avoided.

3.1.7. Offline Connection Set

The Offline Connection Set is a set of messages created to communicate with devices that have failed to go online *(see Section 3.1.6)*, e.g., to allow a new MAC ID to be set. Available materials on this topic go into more detail[3, 9].

3.1.8. Explicit Messaging

All Explicit Messaging in DeviceNet is done via connections and the associated Connection Object instances. However, these objects first must be set up in the device. This can be done by using the Predefined Master/Slave Connection Set to activate a static Connection Object already available in the device or by using the UCMM (Unconnected Message Manager) port of a device, through which this kind of Connection Object can be set up dynamically. The only messages that can be sent to the UCMM are “Open” or “Close” requests that set up or tear down a Messaging Connection, while the only messages that can be sent to the Master/Slave equivalent are “Allocate” or “Release” requests (see also Section 3.1.12). Explicit Messages always pass via the Message Router Object to the individual objects (also refer to Figure 8).

As mentioned in *Section 2.3,* Explicit Messages on DeviceNet have a very compact structure to make them fit into the 8-byte frame in most cases. *Figure 13* shows a typical example of a request message using the 8/8 Message Body Format (1 byte for Class ID, 1 byte for Instance ID).

	Bit number								
Byte offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [0]	Service Code							Message body
2	Class ID								
3	Instance ID								
4 ... 7	Service data ... (optional)								

Figure 13 Non-Fragmented Explicit Request Message Format

The consumer of this Explicit Message responds in the format shown in *Figure 14*. The consumer sets the R/R (Request/Response) bit and repeats the Service Code of the request message. Any data transferred with the response is entered in the service data field.

	Bit number								
Byte offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [1]	Service Code							Message body
2 ... 7	Service data ... (optional)								

Figure 14 Non-Fragmented Explicit Response Message Format

Most messages will use the 8/8 format shown in *Figure 13*, since they only need to address Class and Instance IDs up to 255. If there is a need to address any class/instance combinations above 255, then this is negotiated between the two communication partners during the setup of the connection. Should an error occur, the receiver responds with the Error Response Message. The Service Code for this message is 0x14, and two bytes of error code are returned in the service data field. See endnotes [3, 9] for further details of message encoding, including the use of fragmentation.

3.1.9. I/O Messaging

Since DeviceNet does not use a Real-Time Header or Sequence Count Value like ControlNet and EtherNet/IP do, I/O Messages in DeviceNet have a very compact structure. For I/O data transfers up to 8 bytes long, the data is sent in a non-fragmented message, which uses the entire CAN data field for I/O data. For I/O data transfers longer than 8 bytes, a fragmentation protocol spreads the data over multiple frames. This fragmentation protocol uses one byte of the CAN data field to control the fragmentation of the data. *See Figures 15 and 16* for examples of fragmented and non-fragmented I/O messages. I/O Messages without data (i.e., with zero length data) indicate the "Idle" state of the producing application. Any producing device can do this – master, slave or peer.

	Bit number							
Byte offset	7	6	5	4	3	2	1	0
0 ... 7	Process data (0 – 8 bytes)							

Figure 15 Non-Fragmented I/O Message Format

	Bit number							
Byte offset	7	6	5	4	3	2	1	0
0	Fragmentation protocol							
1 ... 7	Process data (0 – 7 bytes)							

Figure 16 Fragmented I/O Message Format

As mentioned, I/O Messages are used to exchange high-priority application and process data via the network, and this communication is based on the Producer/Consumer model. The associated I/O data are always transferred from one producing application object to one or more consuming application objects. This is undertaken using I/O Messages via I/O Messaging Connection Objects (*Figure 17* shows two consuming applications) that must have been pre-set in the device. This can be done in one of two ways by using:

- The Predefined Master/Slave Connection Set to activate a static I/O Connection Object already available in the device;
- An Explicit Messaging Connection Object already available in the device to dynamically create and set up an appropriate I/O Connection Object.

I/O Messages usually pass directly to the data of the assigned application object. The Assembly Object is the most common application object used with I/O Connections. Also refer to *Figure 8*.

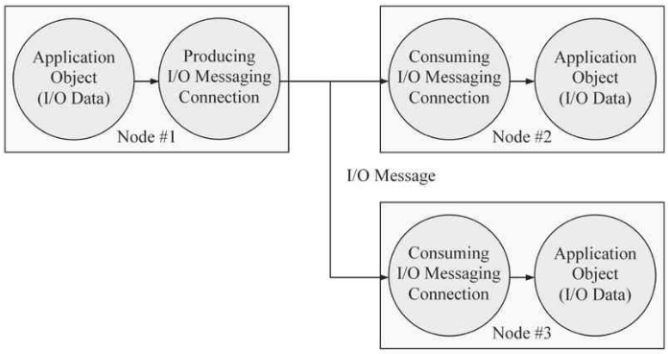


Figure 17 I/O Messaging Connections

3.1.10. Using the CAN Identifier

DeviceNet is based on the standard CAN protocol and therefore uses an 11-bit message identifier. A distinction therefore can be made between 2^{11} = 2048 messages. The 6-bit MAC ID field is sufficient to identify a device because a DeviceNet Network is limited to a maximum of 64 participants.

The overall CAN Identifier range is divided into four Message Groups of varying sizes, as shown in *Figure 18*.

Connection ID = CAN Identifier (bits 10:0)											Used for
10	9	8	7	6	5	4	3	2	1	0	
0	Message ID				Source MAC ID						Message Group 1
1	0	MAC ID						Message ID			Message Group 2
1	1	Message ID			Source MAC ID						Message Group 3
1	1	1	1	1	Message ID						Message Group 4
1	1	1	1	1	1	1	x	x	x	x	Invalid CAN Identifiers

Figure 18 Definition of the Message Groups

In DeviceNet, the CAN Identifier is the Connection ID. This comprises the Message Group ID, the Message ID within this group and the device's MAC ID, which can be the source or destination address. The definition depends on the Message Group and the Message ID. The significance of the message within the system is defined by the Message Group and Message ID.

The four Message Groups are used as follows:

Message Group 1 is assigned 1024 CAN Identifiers (0x0000 – 0x03FF), which is 50 % of all identifiers available. Up to 16 different Message IDs are available per device (node) within this group. The priority of a message from this group is primarily determined by the Message ID (the significance of the message), and only after that by the source MAC ID (the producing device). If two devices transmit at the same time, then the device with a lower Message ID will always win the arbitration. However, if two devices transmit the same Message ID at the same time on the CAN bus, then the device with the lower MAC ID will win. A 16-stage priority system can be set up relatively easily in this manner. The messages of Group 1 are, therefore, well suited for the exchange of high-priority process data.

Message Group 2 is assigned 512 identifiers (0x0400 – 0x05FF). Most of the Message IDs in this group are optionally defined for what is commonly referred to as the Predefined Master/Slave Connection set *(see Section 3.1.12)*. One Message ID is defined for network management *(see Section 3.1.6)*. Priority is determined primarily by the MAC ID and, only after that, by the Message ID. A CAN controller with an 8-bit mask is able to filter out its Group 2 Messages based on MAC ID.

Message Group 3, with 448 CAN Identifiers (0x0600 – 0x07BF), has a similar structure to Message Group 1. Unlike this group, however, low priority process data are mainly exchanged. In addition, the main use of this group is setting up dynamic Explicit Connections. Seven Message IDs per device are possible, and two of these are reserved for what is commonly referred to as the UCMM port *(see Section 3.1.11)*.

Message Group 4, with 48 CAN Identifiers (0x07C0 – 0x07EF), does not include any MAC IDs, only Message IDs. The messages in this group are only used for network management. Four Message IDs are currently assigned for services of the Offline Connection Set.

The remaining 16 CAN Identifiers (0x07F0 – 0x07FF) are invalid CAN IDs and thus are not permitted for use in DeviceNet systems.

With this allocation of CAN Identifiers, the unused CAN Identifiers cannot be used by other devices. Therefore, each device has exactly 16 Message IDs in Group 1, eight Message IDs in Group 2 and seven Message IDs in Group 3. One advantage of this system is that the CAN Identifiers used in the network can always be clearly assigned to a device. Devices are responsible for managing their own identifiers. This simplifies the design, troubleshooting and diagnosis of DeviceNet systems, as a central tool that keeps a record of all assignments on the network is not needed

3.1.11. Connection Establishment

As described in *Sections 3.1.8 and 3.1.9*, messages on DeviceNet are always exchanged in a connection-based manner. Communication objects must be set up for this purpose. These are not initially available when a device is switched on; they first have to be created. The only ports by which a DeviceNet device can be addressed when first switched on are the Unconnected Message Manager port (UCMM port) or the Group 2 Only Unconnected Request port of the Predefined Master/Slave Connection Set. Picture these ports as doors to the device. Only one key will unlock each door. The appropriate key for each lock is the Connection ID – i.e., the CAN Identifier – of the selected port. Other doors in the device can be opened only if and when the appropriate key is available and other instances of Connection Objects are set up.

The setting up of communication relationships (i.e., connections) via the UCMM port represents a general procedure that should be adhered to with all DeviceNet devices. Devices that feature the Predefined Master/Slave Connection Set and are UCMM capable are called Group 2 Servers. A Group 2 Server can be addressed by one or more connections from one or more clients.

Since UCMM capable devices need a good amount of processing power to service multiple communication requests, a simplified communication establishment and I/O data exchange method has been created for low-end devices. This is called the Predefined Master/Slave Connection Set *(see Section 3.1.12)*. This covers as many as five predefined connections that can be activated (assigned) when accessing the device. The Predefined Master/Slave Connection Set represents a subset of the general connection establishment method, and it is limited to pure Master/Slave relations. Slave devices that are not UCMM capable and only support this subset are called Group 2 Only Servers. Only the master that allocates it can address a Group 2 Only Server. All messages received by this device are defined in Message Group 2.

For more details of the connection establishment using UCMM and the Master/Slave Connection Set, refer to endnotes [3, 9].

3.1.12. Predefined Master/Slave Connection Set

Establishing a connection via the UCMM port requires a relatively large number of steps that must be completed to allow data exchange via DeviceNet, and the devices must provide resources to administer the dynamic connections. Because every device can set up a connection with every other device, and the source MAC ID of the devices is contained in the Connection ID, the CAN Identifier (Connection ID) may have to be filtered via software. This depends on how many connections a device supports, and on the type and number of screeners (hardware

CAN ID filters) of the CAN chip used in the device's implementation.

While this approach maximizes use of the multicast, peer-to-peer, and Producer/Consumer capabilities of CAN, a simpler method requiring fewer CPU resources is necessary for low-end devices. This is why a Predefined Master/Slave Connection Set was defined. The Group 2 Only Unconnected Explicit Message port of the Predefined Master/Slave Connection Set provides an interface for a set of five pre-configured connection types in a node.

The basis of this model is a 1:n communication structure consisting of one control device and decentralized I/O devices. The central portion of such a system is known as the "Master" and the decentralized devices are known as "Slaves." Multiple masters are allowed on the network, but a slave can only be allocated to one master at any time.

The predefined Connection Objects occupy instances 1 to 5 in the Connection Object (Class ID 0x05, *see Section 2.4*):

- Explicit Messaging Connection:
  - o Group 2 Explicit Request/Response Message (Instance ID 1)
- I/O Messaging Connections:
  - o Polled I/O Connection (Instance ID 2)
  - o Bit-Strobe I/O Connection (Instance ID 3)
  - o Change of State or Cyclic I/O Connection (Instance ID 4)
  - o Multicast Polling I/O Connection (Instance ID 5)

The messages to the slave are defined in Message Group 2, and some of the responses from the slave are contained in Message Group 1. The distribution of Connection IDs for the Predefined Master/Slave Connection Set is defined as shown in *Figure 19*.

Connection ID = CAN Identifier (bits 10:0)											Used for
10	9	8	7	6	5	4	3	2	1	0	
0	Group 1 Message ID				Source MAC ID						Group 1 Messages
0	1	1	0	0	Source MAC ID						Slave's I/O Multicast Poll Response
0	1	1	0	1	Source MAC ID						Slave's I/O Change of State or Cyclic Message
0	1	1	1	0	Source MAC ID						Slave's I/O Bit-Strobe Response Message
0	1	1	1	1	Source MAC ID						Slave's I/O Poll Response or COS/Cyclic Ack Message
1	0	MAC ID						Group 2 Message ID			Group 2 Messages
1	0	Source MAC ID						0	0	0	Master's I/O Bit-Strobe Command Message
1	0	Source MAC ID						0	0	1	Master's I/O Multicast Poll Group ID
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message
1	0	Source MAC ID						0	1	1	Slave's Explicit/Unconnected Response Messages
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/COS/Cyclic Message
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Messages

Figure 19 Connection IDs of the Predefined Master/Slave Connection Set

Because the CAN ID of most of the messages the master produces contains the destination MAC ID of the slave, it is imperative that only one master talks to any given slave. Therefore, before it can use this Predefined Connection Set, the master must first allocate it with the device. The DeviceNet Object manages this important function in the slave device. It allows only one master to allocate its Predefined Connection Set, thereby preventing duplicate CAN IDs from appearing on the wire.

The two services used are called Allocate_Master/Slave_Connection_Set (Service Code 0x4B) and Release_Group_2_Identifier_Set (Service Code 0x4C). These two services always access instance 1 of the DeviceNet Object (Class ID 0x03) *(see Figure 20)*.

	Bit number								
Byte offset	7	6	5	4	3	2	1	0	
0	Frag [0]	XID	MAC ID						Message header
1	R/R [0]	Service Code [0x4B]							Message body
2…5	Class ID [0x03]								
	Instance ID [0x01]								
	Allocation Choice								
	0	0	Allocator's MAC ID						

Figure 20 Allocate_Master/Slave_Connect_Set Request Message

Figure 20 shows the Allocate Message with 8-bit Class ID and 8-bit Instance ID, a format that is always used when it is sent as a Group 2 Only Unconnected Message. It also may be sent across an existing connection and in a different format if a format other than 8/8 was agreed during the connection establishment.

The Allocation Choice Byte is used to determine which predefined connections are to be allocated *(see Figure 21)*

Bit number							
7	6	5	4	3	2	1	0
Reserved	Ack Suppression	Cyclic	Change of State	Multicast Polling	Bit-Strobe	Polled	Explicit Message

Figure 21 Format of the Allocation Choice Byte

The associated connections are activated by setting the appropriate bits. Change of State and Cyclic are mutually exclusive choices. The Change of State/Cyclic Connection may be configured as not acknowledged using acknowledge suppression. The individual connection types are described in more detail below.

The allocator's MAC ID contains the address of the node (master) that wants to assign the Predefined Master/Slave Connection Set. Byte 0 of this message differs from the allocator's MAC ID if this service has been passed on to a Group 2 Only Server via a Group 2 Only Client (what is commonly referred to as a "proxy function").

The slave, if not already claimed, responds with a Success Message. The connections are now in configuring status. Setting the Expected_Packet_Rate EPR (Set_Attribute_Single service to attribute 9 in the appropriate Connection Object, value in ms) starts the connection's time-monitoring function. The connection then changes into Established State, and I/O Messages begin transferring via this connection.

The allocated connections can be released individually or collectively through the Release_ Group_2_Identifier_Set service (Service Code 0x4C), using the same format as that in *Figure 20,* except that the last byte (Allocator's MAC ID) is omitted.

The following is an explanation of the four I/O Connection types in the Predefined Master/ Slave Connection Set.

3.1.12.1. Polled I/O Connection

A Polled I/O Connection is used to implement a classic Master/Slave relationship between a control unit and a device. In this setup, a master can transfer data to a slave using the Poll Request and receive data from the slave using the Poll Response. *Figure 22* shows the exchange of data between one master and three slaves in Polled I/O mode.

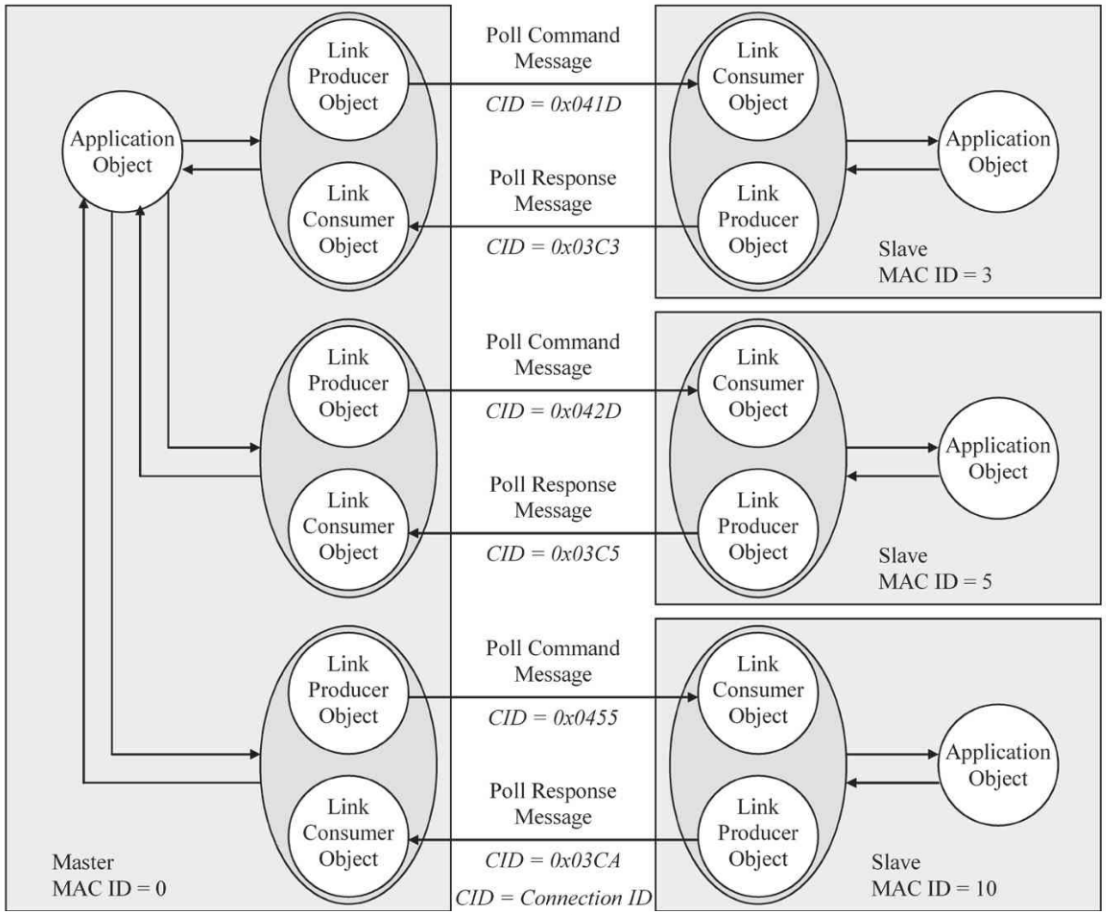


Figure 22 Polled I/O Connections

The amount of data transferred in a message between a master and a slave using the Polled I/O Connection can be any length. If the length exceeds 8 bytes, the fragmentation protocol is automatically used. A Polled I/O Connection is always a point-to-point connection between a master and a slave. The slave consumes the Poll Message and sends back an appropriate response (normally, its input data).

The Polled Connection is subject to a time-monitoring function, which can be adjusted, in the device. A Poll Command must have been received within this time (4 × EPR) or else the connection reverts to time-out mode. When a connection times out, the node optionally may go to a preconfigured fault state as set up by the user. A master usually polls all the slaves in a round-robin manner.

A slave's response time to a poll command is not defined in the DeviceNet Specification. While this provides flexibility for slave devices to be tailored to their primary application, it may also exclude the device from use in higher-speed applications.

3.1.12.2. Bit-Strobe I/O Connection

The master's transmission on this I/O Connection is the Bit-Strobe Command. Using this command, a master multicasts one message to reach all its slaves allocated for the Bit-Strobe Connection. The frame sent by the master using a Bit-Strobe Command is always 8 bytes or 0 bytes (if Idle). From these 8 bytes, each slave is assigned one bit *(see Figure 23)*. Each slave can send back as many as 8 data bytes in its response.

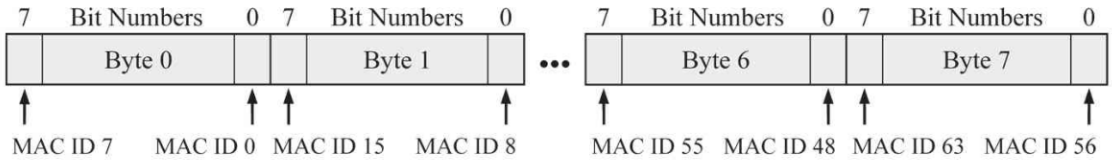


Figure 23 Data Format of the Bit-Strobe I/O Connection

A Bit-Strobe I/O Connection represents a multicast connection between one master and any number of strobe-allocated slaves *(see Figure 24)*. Since all devices in a network receive the Bit-Strobe Command at the same time, they can be synchronized by this command. When the Bit-Strobe Command is received, the slave may consume its associated bit, then send a response of up to 8 bytes.

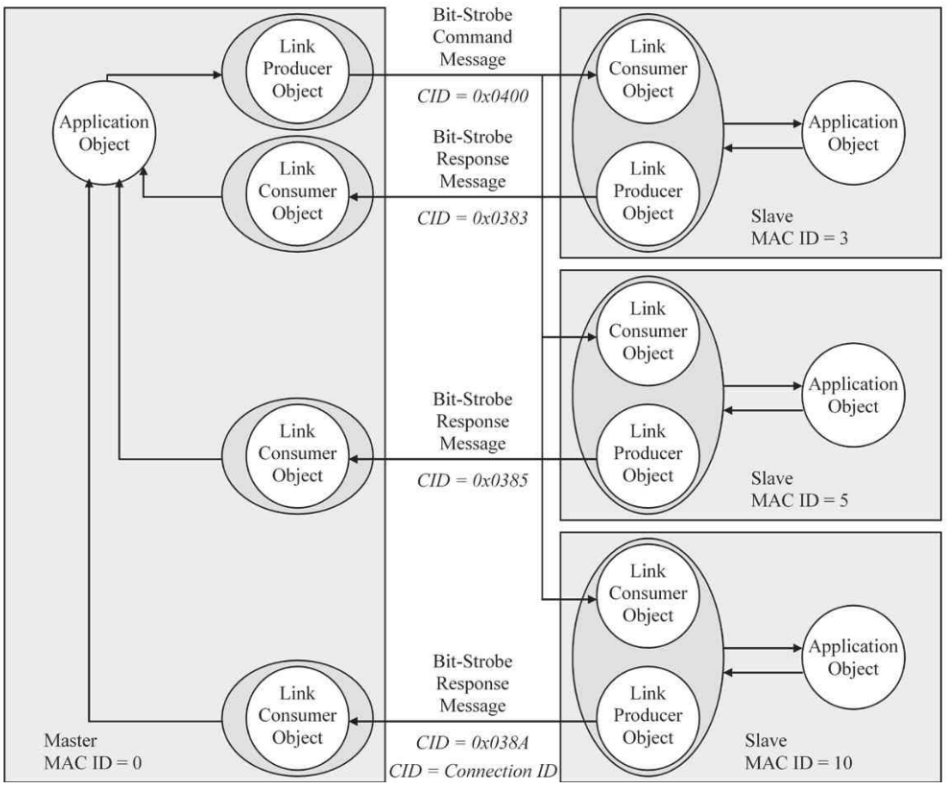


Figure 24 Bit-Strobe I/O Connections

Since this command uses the source MAC ID in the Connection ID *(see Figure 19)*, devices that support the Bit-Strobe I/O Connection and have a CAN chip with screening limited to only 8-bits of the CAN ID (11-bits) must perform software screening of the CAN Identifier.

3.1.12.3. Change of State/Cyclic I/O Connection

The COS/Cyclic I/O Connection differs from the other types of I/O Connections in that both endpoints produce their data independently. This can be done in a change of state or cyclic manner. In the former case, the COS I/O Connection recognizes that the application object data indicated by the Produced_Connection_Path has changed. In the latter case, a timer of the Cyclic I/O Connection expires and therefore triggers the message transfer of the latest data from the application object.

A COS/Cyclic I/O Connection can be set up as acknowledged or unacknowledged. When acknowledged, the consuming side of the connection must define a path to the Acknowledge Handler Object to ensure that the retries, if needed, are properly managed.

Figure 25 shows the various COS/Cyclic I/O Connection possibilities.

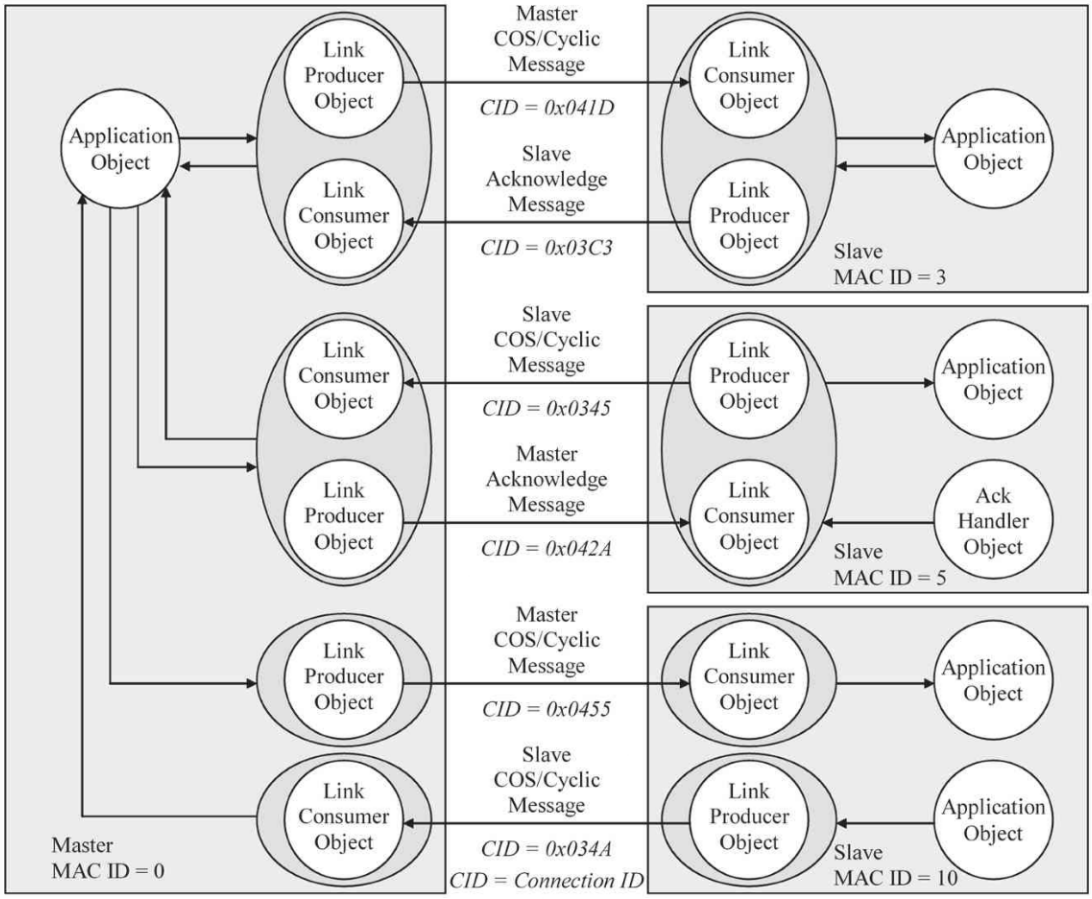


Figure 25 COS/Cyclic I/O Connections

A COS/Cyclic I/O Connection can also originate from a master, making it appear to the slave like a Polled I/O Connection. This can be seen in *Figure 19* since the same Connection ID is issued for the master's Polled I/O Message as is issued for the master's COS/Cyclic I/O Message. COS Connections have two additional behaviors. The Expected Packet Rate (EPR) is used as a default production trigger so that, if the data have not changed after the EPR timer has expired, it will be resent. This "heartbeat," as it is sometimes called, is utilized so the consuming node can know the difference between a "dead" node and one whose data have not changed. COS Connections also have a Production Inhibit Timer feature that prevents a node from producing data too often, and thus using too much network bandwidth. The production inhibit timer determines the amount of time the node must remain quiet after producing data to the network.

3.1.12.4. Multicast Polled I/O Connection

This connection is similar to the regular I/O poll except that all of the slaves belonging to a multicast group consume the same output data from the master. Each slave responds with its own reply data. A unique aspect of this connection is that the master picks the CAN ID from one of the slaves in the multicast group and must then set the consumed CAN ID in each of the other slaves to that same value. If, during runtime, that slave's connection times out, the master must either stop producing its multicast poll command or pick another slave in the group and reset the command CAN ID in all the remaining slaves in the group to that value before sending another Multicast Poll Command.

3.1.12.5. I/O Data Sharing

Due to the inherent broadcast nature of all CAN frames, applications can be set up to "listen" to the data produced by other applications. Such a "listen only" mode is not described in the DeviceNet specification, but some vendors have created products that do exactly that, e.g., "shared inputs" in Allen-Bradley scanners.

3.1.12.6. Typical Master/Slave Start Sequence

Typically, starting up a DeviceNet Network with a scanner and a set of slaves is executed as follows:

- All devices run their self-test sequence and then try to go online with the algorithm described in *Section 3.1.6*. Any device that uses an autobaud mechanism to detect the baud rate of a network has to wait with its Duplicate Node ID Message until it has seen enough CAN frames to detect the correct baud rate
- Once online, slave devices will do nothing until their master allocates them.
- Once online, a master will try to allocate each slave configured into its scan list by running the following sequence of messages:
  - Try to open a connection to the slave using a UCMM Open Message;
  - If successful, the master can then use this connection for further communication with the slave;
  - If not successful, the master will try again after a minimum wait time of one second;
  - If unsuccessful again, the master will try to allocate the slave using the Group 2 Only Unconnected Explicit Request Message (at least for Explicit Messaging) after a minimum wait time of one second;
  - If successful, the master can then use this connection for further communication with the slave;
  - If not successful, the master will try again after a minimum wait time of one second;
  - If unsuccessful again, the master will start over with the UCMM Message after a minimum wait time of one second. This process will carry on indefinitely or until the master has allocated the slave.
- Once the master has allocated the slave, it may carry out some verification to see whether it is safe to start I/O Messaging with the slave. The master also may apply further configuration to the connections it has established, e.g., setting the Explicit Messaging Connection to "Deferred Delete."
- Setting the EPR value(s) brings the I/O Connection(s) to the Established State so that I/O Messaging can commence.

3.1.12.7. Quick Connect Connection Establishment

As of the Edition 1.1 release of Volume 3, DeviceNet also allows an optional method of connection establishment known as Quick Connect. This was designed to provide the same level of protection against duplicate MAC IDs, but to do so in a much shorter time period, allowing connections to be established in a fraction of the time they normally take. This method is useful in applications where nodes are added to an operating network, and the time required for establishing connections directly impacts productivity. For example, in robotic applications, the end-of-arm electronics are often changed out when a new item enters its workspace. These electronics need to be operational very quickly to avoid cycle time delays.

The Quick Connect process includes all the same steps as the typical startup process, but most of them are done in parallel rather than in sequence. As a result, the device self-check and Duplicate MAC ID Check processes begin immediately, and the node goes online almost simultaneously. A failure of the device self-test or a duplicate MAC ID indication causes the device to remove itself from the bus.

In order for applications to benefit fully from this method, Quick Connect must be implemented in both the master and the slave. This feature is selectable through an EDS entry, and, by default, is disabled in nodes that support it.

3.1.12.8. Master/Slave Summary

Device manufacturers can easily support the Predefined Master/Slave Connection Set by using simple BasicCAN controllers. Software screening of the CAN Identifier generally is not necessary, which enables the use of low-cost, 8-bit controllers. This may represent an advantage as far as the devices are concerned but entails some disadvantages for the system design.

Group 2 Only (i.e., UCMM incapable) devices permit only one Explicit Connection between client (master) and server (slave), whereas UCMM-capable devices can maintain Explicit Messaging Connections with more than one client at the same time.

If a device wants to communicate with one of the allocated slaves that does not support UCMM, the master recognizes this situation and sets up a communication relationship with the requestor instead. Any communication between the requestor is then automatically routed via the master. This is called the Proxy function. Since this puts an additional burden on the master and on network bandwidth, it is recommended that slave devices support UCMM.

Although not explicitly defined in the DeviceNet Specification, DeviceNet masters can, under certain conditions, automatically configure their scan lists and/or the devices contained in their scan lists. This functionality simply makes use of the messaging capabilities of masters and slaves that allows the master to read from a slave whatever information is required to start an I/O communication and to download any configurable parameter that has been communicated to the master via EDS. This functionality facilitates the replacement of even complex slave devices without the need for a tool, dramatically reducing system downtime.

3.1.13. Device profiles

DeviceNet devices may utilize any of the CIP profiles. As of the publication date of this book, no DeviceNet-specific profiles are defined in the specification.

3.1.14. Configuration

EDS files for DeviceNet devices can make full use of all EDS features, but they do not necessarily contain all sections. Typical DeviceNet devices contain (apart from the mandatory sections) at least an IO_Info section.

This section specifies which types of Master/Slave connections are supported and which one(s) should be enabled as defaults. It also tells which I/O Connections may be used in parallel.

A full description of what can be done in DeviceNet EDS files would go well beyond the scope of this book. Available materials on this topic that go into more detail[23-24].

3.1.15. Conformance Test

At an early stage, ODVA defined test and approval procedures for DeviceNet devices and systems. Manufacturers are given the opportunity to have their devices checked for conformance with the DeviceNet Specification in one of several independent DeviceNet conformance test centers. Only then do two key characteristics of all DeviceNet devices become possible: interoperability and interchangeability.

Interoperability means that DeviceNet devices from all manufacturers can be configured to operate with each other on the network. **Interchangeability** goes one step further by providing the means for devices of the same type (i.e., they comply with the same Device Profile) to be logical replacements for each other, regardless of the manufacturer.

The conformance test checks both of these characteristics. This test is divided into three parts:

- A **software test** verifies the function of the DeviceNet protocol. Depending on the complexity of the device, up to several thousand messages are transmitted to the device under test (DUT). To ensure a test that is closely adapted to the characteristics of the DUT, the manufacturer must provide a formal description of all relevant features of the DUT.
- A **hardware test** examines conformance with the characteristics of the physical layer. This test checks all requirements of the specification, i.e., mis-wiring protection, overvoltage withstand, grounding, CAN transceiver. The test may be destructive for noncompliant devices.
- A **system interoperability** test verifies that the device can function in a network with more than 60 nodes and with a variety of scanners from various manufacturers.

The software test is available from ODVA. It is a Windows®-based tool, running on various PC CAN interface cards from a number of suppliers. It is recommended that device developers run this test in their own lab before taking devices to the official ODVA test. The hardware test and the system interoperability test involve more complex setups that typically are not available to device developers. When a device passes the test, it is said to be DeviceNet CONFORMANCE TESTED. Many DeviceNet users now demand this seal. A device that has not been tested accordingly has a significant market disadvantage. Devices that have passed conformance testing are published on the ODVA website

3.1.16. Tools

Tools for DeviceNet networks can be divided into three groups:

- Physical layer tools are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;
- Configuration tools are software tools capable of communicating with individual devices for data monitoring and configuration purposes. They can range from very basic software operating on handheld devices to powerful PC-based software packages used to configure complete networks. Most configuration tools are EDS-based; however, more complex devices like scanners tend to have their own configuration applets that are only partially based on EDSs. Some of these tools support multiple access paths to the network, e.g., via Ethernet and suitable routing devices, and thus allow remote access. High-level tools also actively query the devices on the network to identify them and monitor their "health."
- Monitoring tools typically are PC-based software packages that can capture and display CAN frames on the network. A raw CAN frame display may be good enough for some experts, but using a tool that allows both raw CAN display and DeviceNet interpretation of the frames is recommended.

For a typical installation, a configuration tool is all that is needed. However, to ensure that the network is operating reliably, verification with a physical layer tool is highly recommended. Experience shows that the overwhelming majority of DeviceNet network problems are caused by inappropriate physical layer installation. Protocol monitoring tools are used primarily to investigate interoperability problems and to assist during the development process. Turn to the DeviceNet product catalog on the ODVA website to access a list of vendors that provide tools for DeviceNet.

3.1.17. Advice for Developers

Before starting any DeviceNet product development, the following issues should be considered in detail:

- What functionality does the product require today and in future applications?
  - o Slave functionality
  - o Master functionality
  - o Peer-to-peer messaging
  - o Combination of the above
- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?
- What type of hardware should be chosen for this product?
- What kind of firmware should be used for this product? Will a commercially available communication stack be used?
- Will the development of hardware and/or software be done internally or will it be designed by an outside company?
- What are the configuration requirements?
- What design and verification tools should be used?
- What kind of configuration software should be used for this product? Will a commercially available software package be used, i.e. is an EDS adequate to describe the device or is custom software needed?
- Will the product be tested for conformance and interoperability (highly recommended)?
- What is an absolute must before my products can be placed on the market (i.e., own the specification, have a Vendor ID)?

A full discussion of these issues goes well beyond the scope of this book[25].

3.1.18. DeviceNet Summary

Since its introduction in 1994, DeviceNet has been used successfully in millions of nodes in many different applications. It is a *de facto* standard in many countries, which is reflected in several national and international standards[15, 16, 17]. Due to its universal communication characteristics, it is one of the most versatile networks for low-end devices. While optimized for devices with small amounts of I/O, it can easily accommodate larger devices as well. Powerful EDS-based configuration tools allow easy commissioning and configuration of even complex devices without the need to consult manuals.

While most applications are of a Master/Slave type, peer-to-peer communication is used in a rising number of applications, greatly simplifying the design, operation and maintenance of these networks. With the introduction of CIP Safety on DeviceNet, many machine-level applications that previously required a set of dedicated networks today can be accommodated on a single DeviceNet network.

Finally, as a member of the CIP family of networks, DeviceNet can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, just as if they were only one network.

3.2. ControlNet

ControlNet is a deterministic digital communications network that provides high-speed transport of time-critical I/O and messaging data—including upload/download of programming and configuration data and peer-to-peer messaging—on a single physical media link. Each device and/or controller is a node on the network.

ControlNet is a producer/consumer network that supports multiple communication hierarchies and message prioritization. ControlNet systems offer a single point of connection for configuration and control by supporting both implicit (I/O) and explicit messaging. ControlNet's time-based message scheduling mechanism provides network devices with deterministic and predictable access to the network while preventing network collisions. This scheduling mechanism allows time-critical data, which is required on a periodic, repeatable and predictable basis, to be produced on a predefined schedule without the loss of efficiency associated with continuously requesting, or "polling," for the required data.

Like other CIP Networks, ControlNet follows the Open Systems Interconnection (OSI) model, an ISO standard for network communications that is hierarchical in nature. Networks that follow this model define all necessary functions, from physical implementation up to the protocol and methodology to communicate control and information data within and across networks.

Figure 26 shows the relationship between CIP, ControlNet and the ISO/OSI layer model.

The CIP Networks Library contains the ControlNet Adaptation of CIP[4]. All other features are based on CIP

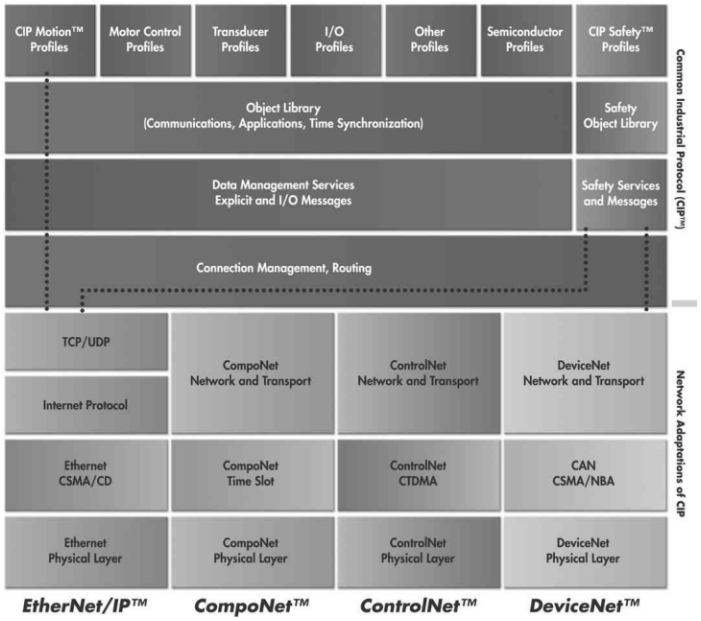


Figure 26 Relationship Between CIP and ControlNet

3.2.1. ControlNet Physical Layer and Frame Structure

The physical layer of ControlNet has been designed specifically for this network; it does not reuse any existing open technology. The basis of the physical layer is a 75Ω coaxial trunk line (typically of RG-6 type cable) terminated at both ends with 75Ω terminating resistors. To reduce impedance mismatch, all ControlNet devices are connected to the network through special taps that consist of a coupling network and a specific length of dropline (1 m). There is no minimum distance requirement between taps, but, since every tap introduces some signal attenuation, each tap reduces the maximum length of the trunkline by 16.3 m. This results in a full length trunkline of 1,000 m with only two taps at the ends while a fully populated physical network with 48 taps allows a trunkline length of 250 m *(see Figure 27)*.

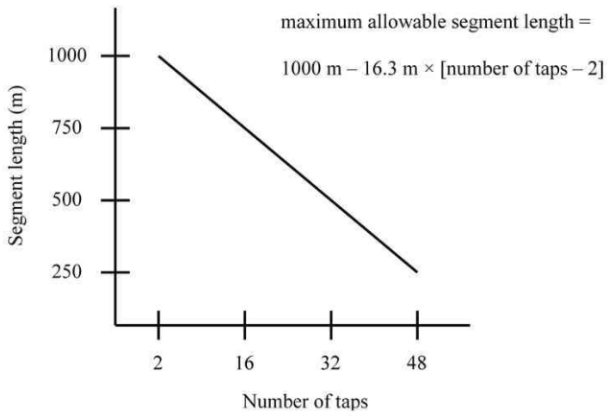


Figure 27 Coax Medium Topology Limits

This physical layer limitation was taken into account from the very beginning by including into the design repeaters that can increase the network size without lowering the speed. Therefore, if a network is to be built with a higher number of nodes (up to 99 nodes are possible) or with a topology that goes beyond the single trunkline limitations, repeaters can be used to extend

the bus. It's possible to create any type of topology: tree, star or linear bus. Even a ring topology is possible using a special type of repeater. Repeaters for fiber optic media can be used either to further increase the system size or to allow isolation of network segments in harsh EMC environments or for high voltage applications.

The number of repeaters between any two nodes was limited to five until recently. Better technology now allows up to 20 repeaters in a series. However, regardless of the media technology used, the overall length of a ControlNet system (the distance between any two nodes on the network) is limited. This fundamental limit is due to propagation delay. With currently available media, this translates into approximately 20 km.

To better accommodate industry requirements, ControlNet supports redundant media, allowing bumpless transfer from primary to secondary media or vice versa if one of them should fail or deteriorate. Developers are encouraged to support this redundant media feature in their designs. For cost-sensitive applications, less expensive device variants may then be created by populating one channel only.

Another feature often used in the process industry is the capability of running ControlNet systems into areas with an explosion hazard. ControlNet is fully approved to meet worldwide standards for intrinsic safety (explosion protection).

Copper media use BNC type connectors. TNC type connectors have been introduced recently for applications that require IP67 protection. Devices also may implement a Network Access Port (NAP). This feature takes advantage of the repeater function of the ControlNet ASICs. It uses an additional connector (RJ-45) with RS 422-based signals that provides easy access to any node on the network for configuration devices.

The signal transmitted on the copper media is a 5-Mbit/second Manchester-encoded signal with an amplitude of up to 9.5 V (pk-pk) at the transmitter that can be attenuated down to 510 mV (pk-pk) at the receiving end. The specification provides reference transmitting and receiving circuits.

3.2.2. Protocol Adaptation

ControlNet can use all features of CIP. The ControlNet frame is big enough that fragmentation is rarely required. Since ControlNet is not used in very simple devices, no scaling is required.

3.2.3. Indicators and Switches

ControlNet devices must be built with Device Status and Network Status indicators as described in the specification. Devices may have additional indicators which must not carry any of the names of those described in the specification.

Devices may be built with or without switches or other directly accessible means for configuration. If switches for the MAC ID exist, then certain rules apply regarding how these values must be used at power up and during the operation of the device.

3.2.4. Additional Objects

Volume 4 defines three additional objects, the ControlNet Object (Class ID = 0xF0), the Keeper Object (Class ID 0xF1) and the Scheduling Object (Class ID 0xF2).

3.2.4.1. ControlNet Object (Class ID = 0xF0)

The ControlNet Object contains a host of information about the state of the device's ControlNet interface, among them diagnostic counters, data link and timing parameters and the MAC ID. A ControlNet Object is required for every physical layer attachment of the device. A redundant channel pair counts as one attachment.

3.2.4.2. Keeper Object (Class ID = 0xF1)

The Keeper Object (not required for every device) holds (for the network scheduling software) a copy of the Connection Originator schedule data for all Connection Originator devices using the network. Every ControlNet Network with scheduled I/O traffic must have at least one device with a Keeper Object (typically, a PLC or another Connection Originator). If there are multiple Keeper Objects on a network, they perform negotiations to determine which Keeper is the Master Keeper and which Keeper(s) perform Backup Keeper responsibilities. The Master Keeper is the Keeper actively distributing attributes to the nodes on the network. A Backup Keeper is one that monitors Keeper-related network activity and can transition into the role of Master Keeper should the original Master Keeper become inoperable.

3.2.4.3. Scheduling Object (Class ID = 0xF2)

The Scheduling Object is required in every device that can originate an I/O Messaging Connection. Whenever a network scheduling tool accesses a Connection Originator on a ControlNet Network, an instance of the Scheduling Object is created and a set of object-specific services is used to interface with this object. Once the instance is created, the network scheduling tool can then read and write connection data for all connections that originate from this device. After having read the connection data from all Connection Originators, the network scheduling tool can calculate an overall schedule for the ControlNet Network and write this data back to all Connection Originators. The scheduling session is ended by deleting the instance of the Scheduling Object.

3.2.5. Network Access

ControlNet's bus access mechanism allows full determinism and repeatability while still maintaining sufficient flexibility for various I/O Message triggers and Explicit Messaging. This bus access mechanism is called Concurrent Time Domain Multiple Access (CTDMA); it is illustrated in *Figure 28*.

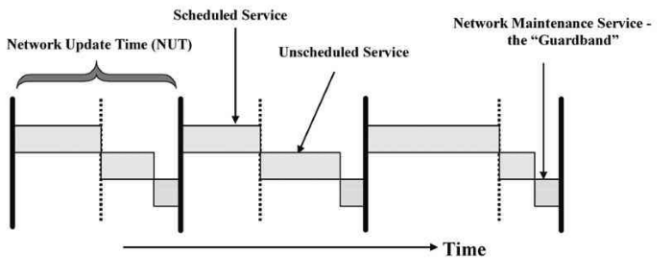


Figure 28 Media Access through CTDMA (Concurrent Time Domain Multiple Access)

The time axis is divided into equal intervals called Network Update Time (NUT). Each NUT is subdivided into a Scheduled Service Time, an Unscheduled Service Time and a Guardband Time.

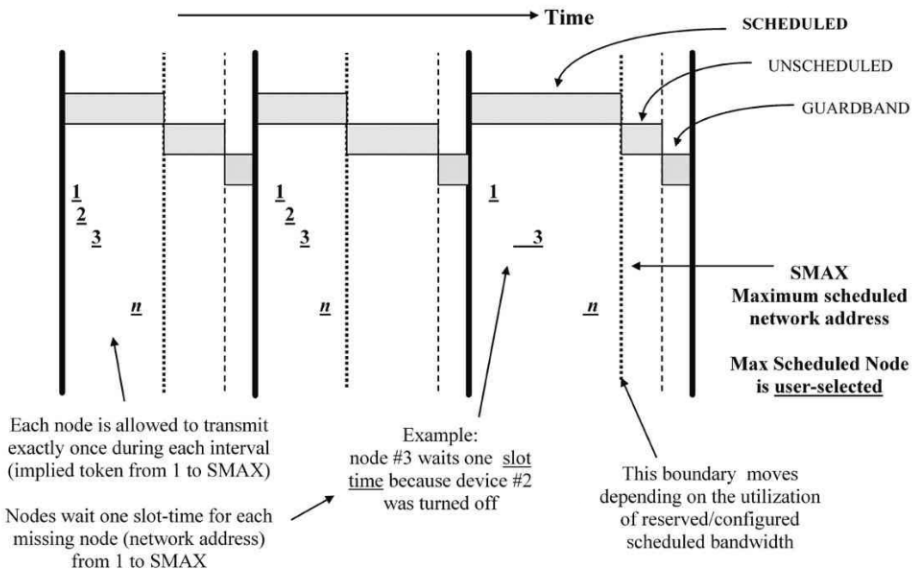


Figure 29 Scheduled Service

Figure 29 shows the function of the Scheduled Service. Every node up to, and including, the SMAX node (maximum node number participating in the Scheduled Service) has a chance to send a message within the Scheduled Service. If a particular node has no data to send, it will nevertheless send a short frame to indicate that it is still alive. If a node fails to send its frame, the next-higher node number will step in after a very short, predetermined waiting time. This process ensures that a node failure will not lead to an interruption of the NUT cycle.

Figure 30 shows the function of the Unscheduled Service. Since this service is designed for non-time-critical messages, only one node is guaranteed access to the bus during the Unscheduled Service Time. If there is time left, the other nodes (with higher node numbers) will also get a chance to send. As with the Scheduled Service Time, if a node fails to send during its turn, the next node will step in. The node number that is allowed to send first within the Unscheduled Service Time is increased by one in each NUT. This guarantees an equal chance to all nodes. All node sequencing in this interval wraps; UMAX is followed by the lowest node number (typically = 1) on the network.

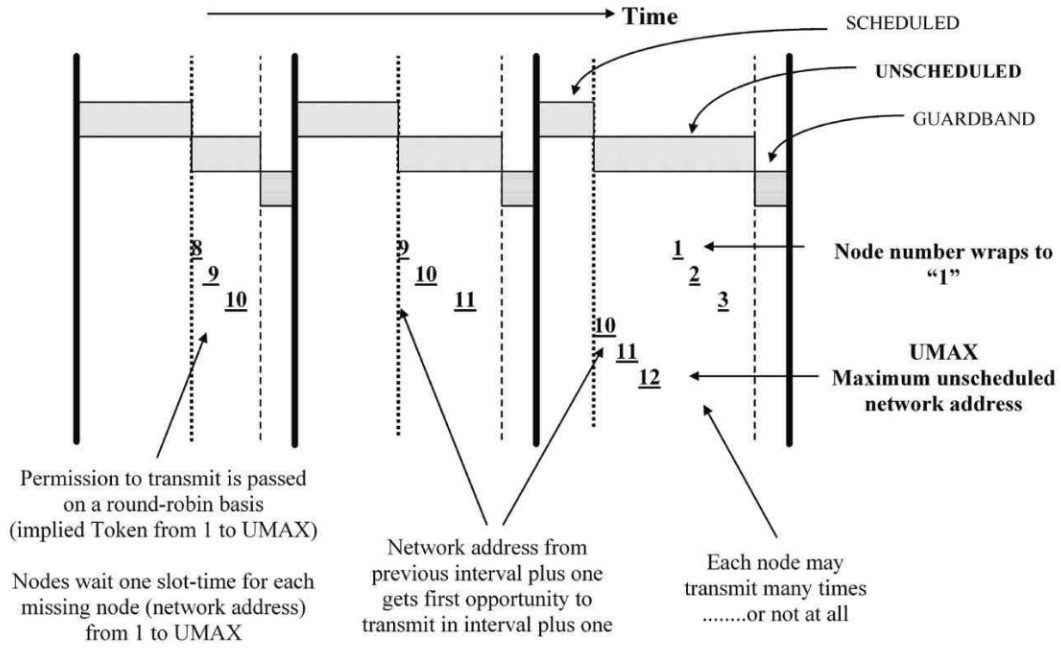


Figure 30 Unscheduled Service

These two service intervals, combined with the Guardband, guarantee determinism and repeatability while still maintaining sufficient freedom to allow for unscheduled message transmissions, e.g., for parameterization.

3.2.6. Frame Description

Every frame transmitted on ControlNet has the format of the MAC frame, see *Figure 31*.

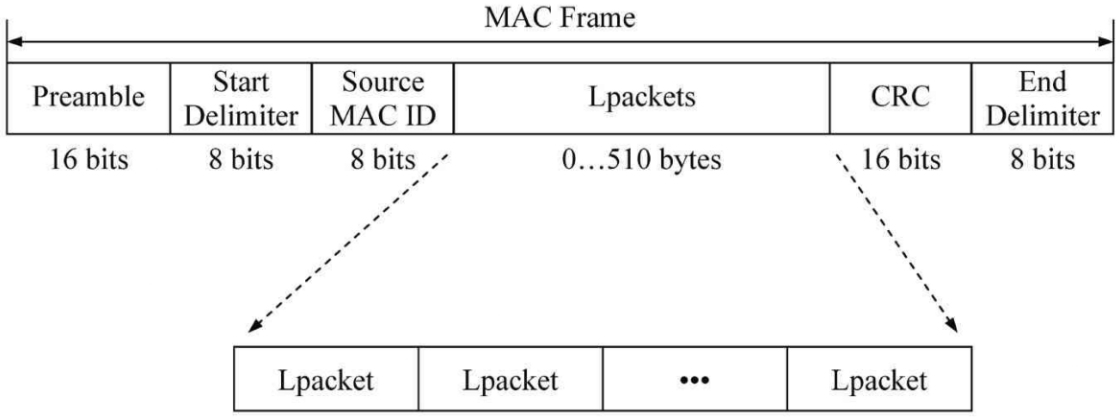


Figure 31 MAC Frame Format

Within every MAC frame, a field of up to 510 bytes is available for transmitting data or messages. This field may be populated with one or several Lpackets (link packets). These Lpackets carry the individual CIP messages (I/O or Explicit). Specialized Lpackets are used for network management. Since all nodes always listens to all MAC frames, they have no problem consuming any of the Lpackets in a frame that is unicast, multicast or broadcast in nature. This feature allows fine-tuned multicasting of small amounts of data to different sets of consumers without too much overhead.

There are two types of Lpacket formats, fixed tag and generic tag. The fixed tag Lpackets are used for Unconnected Messaging and network administration while the generic tag Lpackets are used for all Connected Messaging (I/O and Explicit).

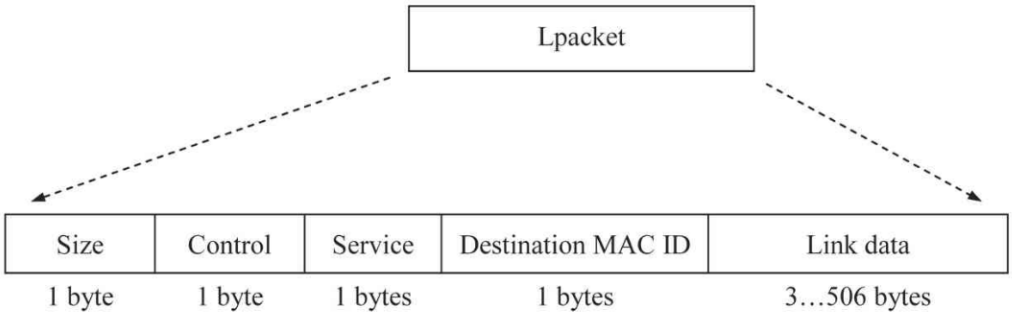


Figure 32 Fixed Tag Lpacket Format

Figure 32 shows the format of a fixed tag Lpacket. By including the destination MAC ID, this format reflects the fact that these Lpackets are always directed from the requesting device (sending the MAC frame) to the target device (the destination MAC ID). The service byte within a fixed tag Lpacket does not represent the service of an Explicit Message, but a more general service type, since the fixed tag Lpacket format can be used for a variety of actions, such as network administration.

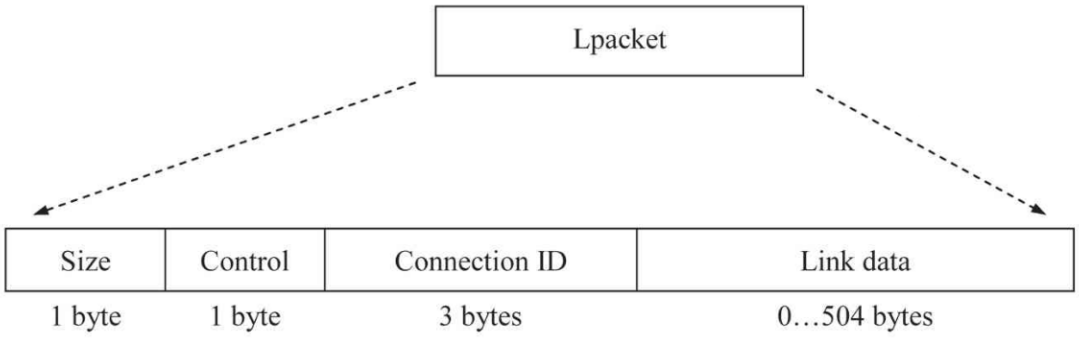


Figure 33 Generic Tag Lpacket Format

Figure 33 shows the format of a generic tag Lpacket. The size byte specifies the number of words within the Lpacket, while the control byte gives information on what type of Lpacket this is. The 3 byte Connection Identifier specifies which connection this Lpacket belongs to. These three bytes are the three lower bytes of the 4 byte Connection ID specified in the Forward_Open message; the uppermost byte is always zero. For a device that receives the MAC frame, the Connection ID indicates whether to ignore the Lpacket (the device is not part of the connection), to consume the data and forward it to the application (the device is an end point of this connection) or to forward the data to another network (the device acts as a router in a routed connection).

3.2.7. Network Startup

After power-on, every ControlNet device goes through a process of accessing the ControlNet communication network and learning the current NUT and other timing requirements. This is a fairly complex process typically handled by commercially available ControlNet ASICs. It is beyond the scope of this book to describe the details here.

3.2.8. Explicit Messaging

Unlike DeviceNet, Explicit Messages on ControlNet can be sent either connected or unconnected; both are transmitted within the unscheduled part of the NUT. Connected Explicit Messaging first requires setting up a connection *(see Section 3.2.10.)*. This, of course, means that all resources required for managing the connection are reserved for this purpose as long as the connection exists, which allows more timely responses to message requests. This is very useful when the application requires periodic explicit requests. Most Explicit Messages also can be sent unconnected, but this mechanism makes use of generally limited resources in nodes that sometimes can be highly utilized. For this reason, unconnected messaging should be used only when the application requires very irregular and infrequent request intervals. Every part of an Explicit Message (request, response, acknowledgements) is wrapped into an Lpacket using the fixed tag Lpacket format for Unconnected Messaging *(see Figure 32)* and the generic tag Lpacket format for Connected Messaging *(see Figure 33*). The service/class/instance/attribute fields *(see Section 2.3.)* of the Explicit Message are contained in the link data field.

3.2.9. I/O Messaging

ControlNet I/O Messaging, like any other CIP I/O Messaging, is done across connections, and it always takes place in the scheduled part of the NUT. Only one MAC frame may be transmitted by any device within its time slot, but this MAC frame may contain multiple Lpackets so that data can be sent to multiple nodes in one NUT. The individual Lpackets may be consumed by one node only or by multiple nodes if they are set up to consume the same data.

I/O Messages use the generic tag Lpacket format *(see Figure 33)*. The link data field contains the I/O data prefixed with a 16-bit Sequence Count Number for the packet. I/O data transmission without the Sequence Count Number is possible in principle, but is not used today. Run/Idle can be indicated within a prefixed Real-Time Header or by sending the data packet (Run) or no data packet (Idle). The method used is indicated in the connection parameters of the Connection Manager section of the EDS. However, only the Real-Time Header method has been used for ControlNet up to now.

3.2.10. Connection Establishment

All connections on ControlNet are established using a UCMM Forward_Open message *(see Section 2.3.)*; therefore, all devices that support Connected Messaging must support the UCMM function.

3.2.11. Device Classes

Four classes of device functionality are built with CIP. While they are not explicitly defined in the specification, they are useful for distinguishing among several classes of devices. The four classes are described here.

- The minimal device function is that of a **Messaging Server**, which is used for Explicit Messaging applications only and acts as a target for Connected and Unconnected Explicit Messages, e.g., for program upload/download, data collection, status monitoring, etc.;
- The next device class is an **I/O Server**, which adds I/O Messaging Support to a Messaging Server device and acts as a target for both Explicit and I/O Messages, e.g., simple I/O Devices, Pneumatic Valves, AC Drives. These devices are also called Adapters;
- Another device class is a **Messaging Client**, which adds client support to Messaging Server applications and acts as a target and as an originator for messaging applications, e.g., computer interface cards, HMI devices;
- The most powerful type of device is an **I/O Scanner,** which adds I/O Message origination support to the functionality of all the other device classes, and which acts as a target and as an originator for Explicit and I/O Messages, e.g., PLCs, I/O Scanners.

3.2.12. Device Profiles

ControlNet devices may utilize any of the profiles defined by CIP. As of the publication date of this book, no ControlNet-specific profiles have been defined.

3.2.13. Configuration

ControlNet devices typically come with Electronic Data Sheets (EDS) as described in *Section 2.7*. For EDS-based configuration tools, the EDS should contain a Connection Manager section to describe the details of the connections that can be made into the device. This section basically mirrors the contents of the Forward_Open message that a Connection Originator would send to the device. Multiple connections can be specified within an EDS, then one or more can be chosen by the configuration tool.

An EDS may also contain individual parameters and/or a Configuration Assembly with a complete description of all parameters within this Assembly. In many applications, the Configuration Assembly is transmitted as an attachment to the Forward_Open message.

3.2.14. Conformance Test

ControlNet International has defined a conformance test for ControlNet devices. Currently, this test is a protocol conformance test only since it is expected that most implementations use the commercially available components for transformers and drivers. The software test is available from ControlNet International. It is a Windows®-based tool that runs on a PC interface card through a NAP connection *(see Section 3.2.1.)*. For more information on Conformance Testing, see *Section 6.*

3.2.15. Tools

Tools for ControlNet Networks can be divided into three groups:

- Physical layer tools are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;
- Configuration tools are software tools capable of communicating with individual devices for data monitoring and configuration purposes. Most configuration tools are EDS-based; however, more complex devices like scanners tend to have their own configuration applets that are only partially based on EDSs. Some of these tools support multiple access paths to the network, e.g., via Ethernet and suitable routing devices, and thus allow remote access. High-level tools also actively query the devices on the network to identify them and monitor their health. Configuration tools also may be integrated into other packages like PLC programming software.
- Monitoring tools typically are PC-based software packages that can capture and display the ControlNet frames on the network. A raw ControlNet frame display may be good enough in some instances, but using a tool that can display both raw ControlNet frames and interpreted frames is recommended.

For a typical installation, a configuration tool is all that is needed. However, to ensure the network is operating reliably, testing with a physical layer tool is highly recommended. Experience shows that the overwhelming majority of ControlNet Network problems are caused by inappropriate physical layer installation. Protocol monitoring tools are mainly used to investigate interoperability problems and to assist during the development process.

Turn to the ControlNet product catalog on the ControlNet International website to access a list of vendors that provide tools for ControlNet

3.2.16. Advice for Developers

Before any development of a ControlNet product is started, the following issues should be considered in detail:

- What functionality (Device Classes, see *Section 3.2.11.*) does the product require today and in future applications?
    - o Messaging server only
    - o Adapter functionality
    - o Messaging client
    - o Scanner functionality
- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?
- Will the development be based on commercially available hardware components and software packages (recommended) or designed from scratch (possible but costly)?
- What are the configuration requirements?
- Will the product be tested for conformance (highly recommended)?
- What design and verification tools should be used?
- What is an absolute must before products can be placed on the market (own the specification, have a Vendor ID)?

Turn to the ControlNet International website for a list of companies that can support ControlNet developments.

3.2.17. ControlNet Summary

Since its introduction in 1997, ControlNet has been used successfully in hundreds of thousands of nodes in many different applications. It is the network of choice for many high speed I/O and PLC interlocking applications. Like DeviceNet, ControlNet has become an international standard[18]. Due to its universal communication characteristics, it is one of the most powerful controller-level networks available.

ControlNet's greatest strengths are its full determinism and repeatability. These strengths make it ideally suited for many high-speed applications, in which ControlNet maintains full Explicit Messaging capabilities without compromising its real-time behavior.

Finally, as a member of the CIP family of networks, ControlNet can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, just as if they were only one network.

3.3. EtherNet/IP

EtherNet/IP, a technology supported by both ODVA and ControlNet International, is the newest member of the CIP Family. Using CIP as its upper-layer protocol, EtherNet/IP extends the application of Ethernet TCP/IP to the plant floor. This enables all EtherNet/IP network devices to speak the same language. EtherNet/IP can coexist with any other protocol running on top of the standard TCP/UDP Transport Layer, and with other CIP Networks. EtherNet/IP—CIP plus Internet and Ethernet standards—provides a pure, standards-based Ethernet solution for interoperability among manufacturing enterprise networks, and it enables Internet and enterprise connectivity anywhere, anytime.

Due to the length of Ethernet frames and the typical multi-master structure of Ethernet networks, there are no particular limitations in the EtherNet/IP implementation of CIP. Basically, all that is required is a mechanism to encode CIP messages into Ethernet frames

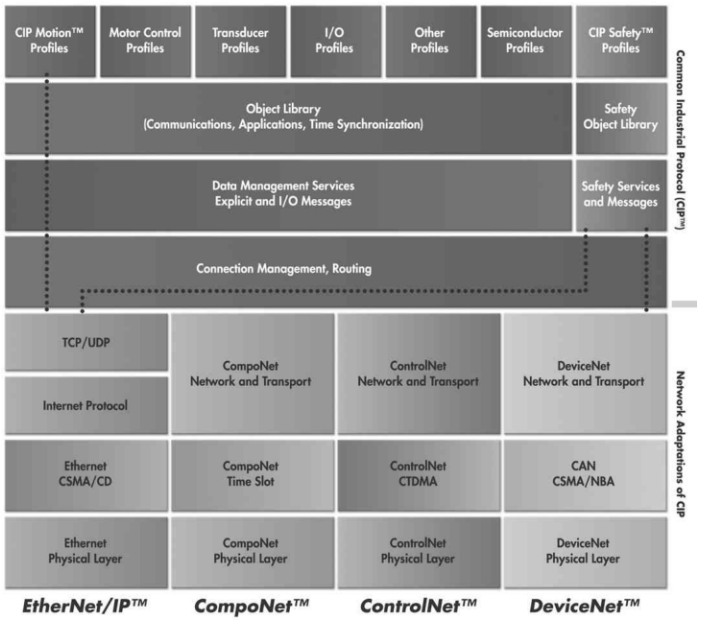


Figure 34 Relationship Between CIP and EtherNet/IP

Volume 2 of The CIP Networks Library is the EtherNet/IP Adaptation of CIP. This volume defines how CIP is adapted for use on Ethernet. An encapsulation mechanism *(see Section 3.3.7.)* is defined for EtherNet/IP specifying how I/O and Explicit Messages are carried in Ethernet frames. The well-known TCP/IP protocol is used for encapsulating Explicit Messages, while UDP/IP is used for encapsulating I/O Messages. Since the commonly implemented TCP/IP and UDP/IP protocol stacks are used for encapsulation, many applications will not require extra middleware for this purpose.

Ethernet has its roots in the office computing environment, which is not traditionally concerned with determinism like industrial applications are. However, with the proper selection and configuration of infrastructure devices *(see Section 3.3.16.)* and the use of fast data rates with full duplex communications, Ethernet's level of determinism is more than adequate for use in industrial control applications. Additionally, extensions to CIP like CIP Sync *(see Section 5.1.)* allow EtherNet/IP to be used in highly synchronous and deterministic applications like coordinated drives and motion control.

3.3.1. Physical Layer Adaptation

Since EtherNet/IP takes the Ethernet protocol to the factory floor, recommendations are made in Volume 2[2] regarding grounding, isolation and cable and connector construction that are designed to make EtherNet/IP successful in a typical factory automation environment. These changes do not affect the actual signaling or interoperability with standard Ethernet products, but simply make devices more suitable for industrial environments. As a result, two levels of performance criteria are defined:

- The **COTS (Commercial Off The Shelf) EtherNet/IP** Level provides basic Ethernet connectivity. This level includes the well-known RJ-45 type Ethernet connector but specifies topology constraints (e.g., up to 100m) and cabling requirements through references to specific IEEE, ANSI/TIA/EIA standards. Such devices are typically suited for IP20 applications.
- The **Industrial EtherNet/IP Level** goes beyond the COTS Level by specifying minimum environmental, cabling and connector requirements that include IEC, ANSI/TIA/EIA standards. Connectors required for the Industrial EtherNet/IP Level include a sealed RJ-45 connector as well as a more compact, D-coded M12-4 connector. Both connectors can achieve an IP67 rating

Cat 5E or Cat 6 shielded or unshielded cables are recommended for EtherNet/IP. The use of shielded cables is specifically recommended in applications where adjacent material, such as metal cable ducts, may have substantial influence on the characteristics of the cable. Copper media may be used only for distances up to 100 m. Fiber-optic media are recommended for longer distances. Fiber-optic media may also be advisable for applications with very high electromagnetic disturbances or high-voltage potential differences between devices

3.3.2. Frame Structure

EtherNet/IP uses standard Ethernet TCP/IP and UDP/IP frames as defined by international standards[11, 26, 27, 29]. Therefore, no further frame details are described here.

3.3.3. Protocol Adaptation

EtherNet/IP can use all features of CIP. The Ethernet frame is big enough that fragmentation is rarely required. Since EtherNet/IP is not expected to be used in very simple devices, no further scaling than that described in *Section 3.3.10.* is required.

3.3.4. Indicators and Switches

EtherNet/IP devices that need to conform to the Industrial EtherNet/IP Level must have the set of indicators set forth in the specification. Devices may have additional indicators which must not carry any of the names of those described in the specification.

Devices may be built with or without switches or other directly accessible means for configuration.

3.3.5. Additional Objects

Volume 2 defines two additional objects, the TCP/IP Object (Class ID = 0xF5) and the Ethernet Link Object (Class ID 0xF6) that are found only on EtherNet/IP devices.

3.3.5.1. TCP/IP Object (Class ID 0xF5)

The TCP/IP Interface Object provides a mechanism for configuring a device's TCP/IP network interface. Examples of configurable items include the device's IP address, network mask and gateway address.

3.3.5.2. Ethernet Link Object (Class ID 0xF6)

The Ethernet Link Object maintains specific counters and status information for the Ethernet 802.3 communications interface. Each device has exactly one instance of the Ethernet Link Object for each Ethernet 802.3 communications interface. A request to access instance 1 of the Ethernet Link Object always refers to the instance associated with the communications interface over which the request was received.

3.3.6. IP Address Assignment

Since the initial development of TCP/IP, numerous methods for configuring a device's IP address have evolved. Not all of these methods are suitable for industrial control devices. In the office environment, for example, it is common for a PC to obtain its IP address via DHCP (Dynamic Host Configuration Protocol), meaning that it can potentially acquire a different address each time the PC reboots. This is acceptable because the PC is typically a client device that only makes requests, so there is no impact if its IP address changes.

However, for an industrial control device that is a target of communication requests, the IP address cannot change at each power up. A PLC, for example, must be at the same address each time it powers up.

To further complicate matters, the only interface common to all EtherNet/IP devices is an Ethernet communications port. Some devices may also have a serial port, a user interface display, hardware switches or other interfaces, but these are not universally shared across all devices. Since Ethernet is the common interface, the initial IP address must at least be configurable over Ethernet.

The EtherNet/IP Specification, via the TCP/IP Interface Object, defines a number of ways to configure a device's IP address. A device may obtain its IP address via BOOTP (Bootstrap Protocol), via DHCP or via an explicit Set_Attribute (single or set-all) service. None of these methods is mandated however. As a result, vendors could choose different methods for configuring IP addresses.

From the user's perspective, it is desirable for vendors to support some common mechanism(s) for IP address configuration. The current ODVA recommendations on this subject can be downloaded from the ODVA website[7].

3.3.7. EtherNet/IP Encapsulation

EtherNet/IP is based entirely on existing TCP/IP and UPD/IP technologies and uses these principles without any modification. TCP/IP is mainly used for the transmission of Explicit Messages while UDP/IP is used mainly for I/O Messaging.

The encapsulation protocol defines two reserved TCP port numbers. All EtherNet/IP devices accept at least two TCP connections on TCP port number 0xAF12. This port is used for all TCP-based Explicit Messaging, either connected or unconnected. It is also used for the encapsulation protocol commands that are employed when setting up communications between nodes. Some encapsulation commands may also be sent to port 0xAF12 via UDP datagrams.

Port 0x08AE is used by any devices that support EtherNet/IP's UDP-based I/O messaging in order to take advantage of the multicast capabilities of IP. Multicast data flow makes more efficient use of the available bandwidth and provides for better data consistency across the system. Being connectionless, UDP is well suited to this purpose. Unlike TCP, UDP does not have the ability to reorder packets. Therefore, whenever UDP is used to send an encapsulated message, the entire message is sent in a single UDP packet, and only one encapsulated message is present in any UDP packet.

3.3.7.1. General Use of the Ethernet frame

Since EtherNet/IP is completely based on Ethernet with TCP/IP and UDP/IP, all CIP-related messages sent on an EtherNet/IP Network are based on Ethernet frames with an IP header *(see Figure 35).*

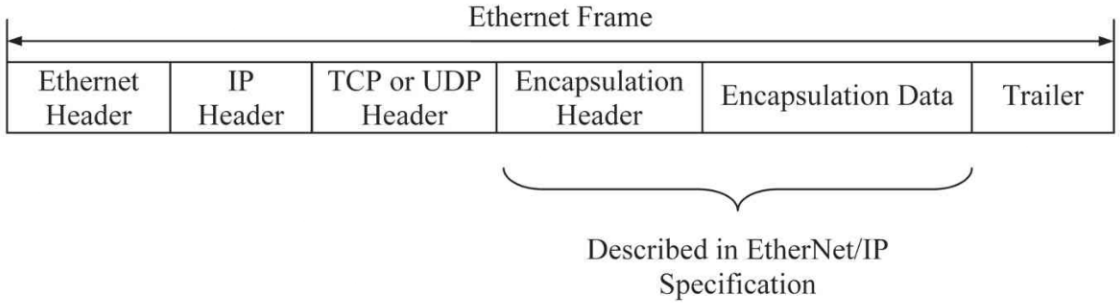


Figure 35 Relationship Between CIP and Ethernet Frames

The Ethernet header, the IP header and the TCP or UDP headers are described through international standards *(see Section 3.3.2.)*; therefore, details of these headers are mentioned only in the EtherNet/IP Specification when it is necessary to understand how they are used.

The encapsulation header is a description of the meaning of the encapsulation data. Most encapsulation data use the so-called Common Packet Format. I/O Messages sent in UDP frames do not carry an encapsulation header, but they still follow the Common Packet Format.

3.3.7.2. Encapsulation Header and Encapsulation Commands

The overall encapsulation packet has the structure described in *Figure 36*.

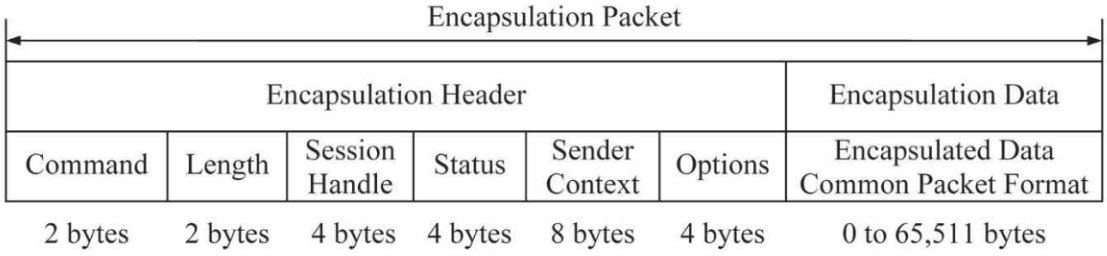


Figure 36 Structure of the Encapsulation Packet

While the description of some of the encapsulation header details would go beyond the scope of this book, the command field requires more attention here. However, only those commands that are needed to understand the EtherNet/IP protocol are described, and their description only lists the main features. The encapsulated data as such follows the Common Packet Format *(see Section 3.3.7.2.4.)*.

3.3.7.2.1. ListIdentity Command

The ListIdentity command typically is sent as a broadcast UDP message to tell all EtherNet/IP devices to return a data set with identity information. This command typically is used by software tools to browse a network.

3.3.7.2.2. RegisterSession/UnRegisterSession Commands

These two commands are used to open and close an Encapsulation Session between two devices. Once such a session is established, it can be used to exchange further messages. Multiple sessions may exist between two devices, but this is not common.

The device requesting the session creates a sender context value, and the device receiving the session request creates a session handle. Both values are used to identify messages between the two devices that use this session.

3.3.7.2.3. SendRRData/SendUnitData Commands

The SendRRData Command is used for Unconnected Messaging, and the SendUnitData Command is used for Connected Explicit Messaging.

3.3.7.2.4. Common Packet Format

The Common Packet Format (CPF) is a construct that provides a way to structure the Encapsulation Data field for those Encapsulation commands that specify Encapsulation data. If the command definition requires it, the CPF allows packing of multiple items into one encapsulation frame, as shown in *Figure 37*.

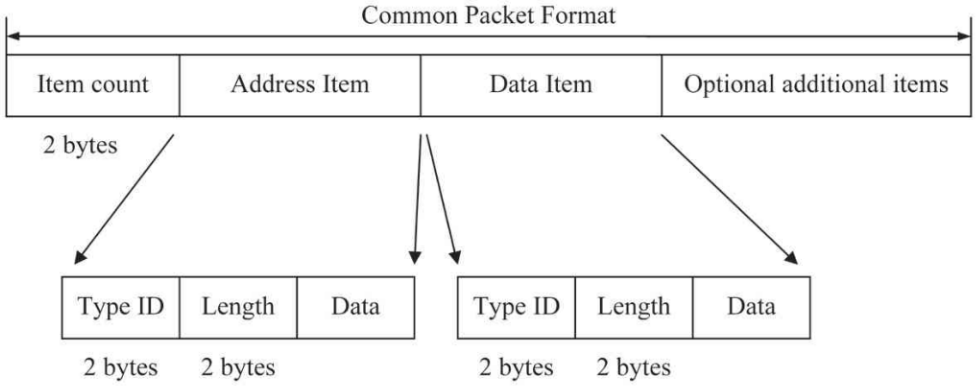


Figure 37 Example of the Common Packet Format

All encapsulated messages are then assembled using at least these two items within the Common Packet Format.

3.3.8. Use of the Encapsulation Data

3.3.8.1. Explicit Messaging

Explicit Messages on EtherNet/IP, unlike those on DeviceNet, can be sent either connected or unconnected. Connected Explicit Messaging requires setting up a connection first *(see Section 3.3.9.)*. This means that all resources required for managing the connection are reserved for this purpose as long as the connection exists, which provides for more timely responses to message requests. This is very useful in applications that require periodic explicit requests. Most Explicit Messages also can be sent unconnected, but this mechanism makes use of generally limited resources in nodes that sometimes can be highly utilized. For this reason, unconnected messaging should be used only when the application requires very irregular and infrequent request intervals. Explicit Messages on EtherNet/IP are sent with a TCP/IP header and use the SendRRData Encapsulation Command (unconnected) and the SendUnitData Encapsulation Command (connected). As an example, the full encapsulation of a UCMM request is shown in *Figure 38*.

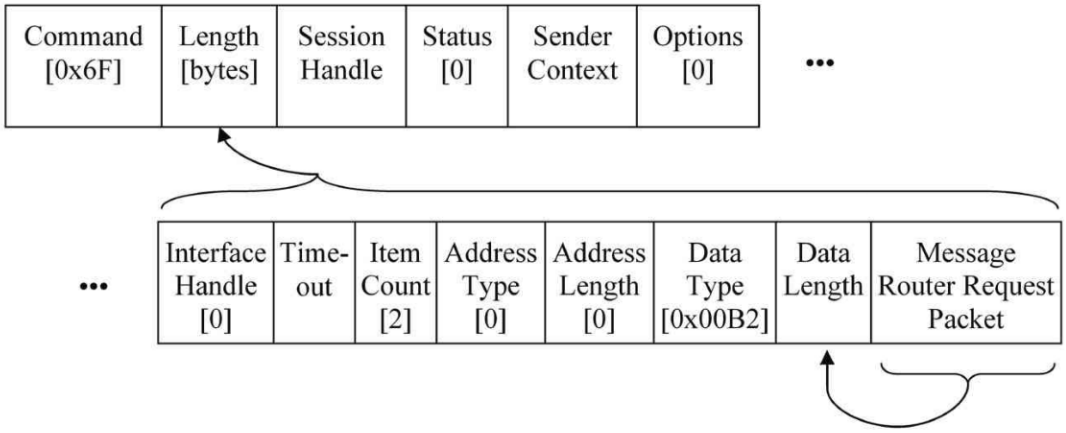


Figure 38 UCMM Request Encapsulation

The Message Router Request Packet, containing the message as such, follows the general format of Explicit Messages defined by CIP.

3.3.8.2. I/O Messaging

I/O Messages on EtherNet/IP are sent with a UDP/IP header. No encapsulation header is required, but the message still follows the Common Packet Format, see *Figure 39* for an example.

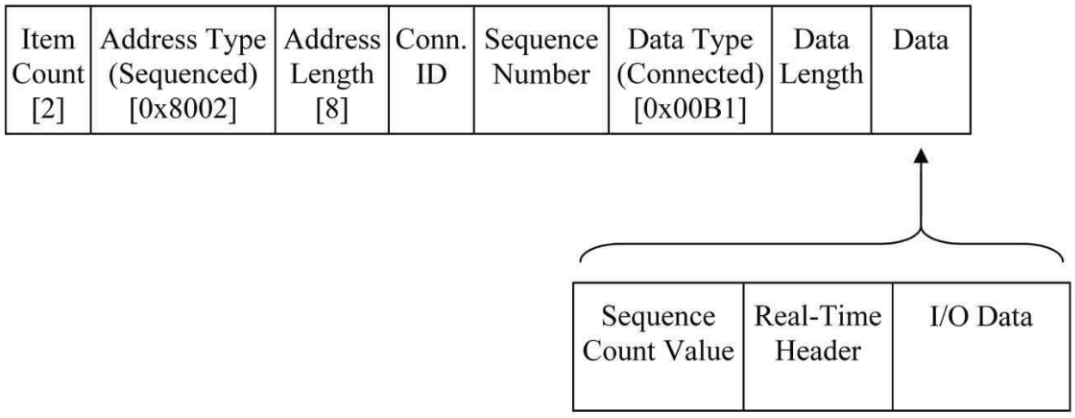


Figure 39 I/O Message Encapsulation

The data field contains the I/O data prefixed with a 16-bit Sequence Count Value for the packet. I/O data transmission without the Sequence Count Value is possible in principle, but is not used today. Run/Idle can be indicated within a Real-Time Header or by sending the data packet (Run) or no data packet (Idle). The method used is indicated in the connection parameters of the Connection Manager section of the EDS. However, the Real-Time Header method is recommended for use in EtherNet/IP and this is what is shown in *Figure 39*.

I/O Messages from the originator to the target are typically sent as UDP unicast frames, while those sent from the target to the originator are typically sent as UDP multicast frames. This allows other EtherNet/IP devices to listen to the input data. To avoid these UDP multicast frames propagating all over the network, the use of switches that support IGMP Snooping is highly recommended. IGMP (Internet Group Management Protocol[39]) is a

protocol that allows the automatic creation of multicast groups. Using this functionality, the switch will automatically create and maintain a multicast group consisting of the devices that need to consume these multicast messages. Once the multicast groups have been established, the switch will direct such messages only to those devices that have subscribed to the multicast group of that message.

3.3.9. Connection Establishment

All connections on EtherNet/IP are established using a UCMM Forward_Open message *(see Section 2.3.);* therefore all devices that support Connected Messaging must support the UCMM function.

3.3.10. Device Classes

Four classes of device functionality are built with CIP. While they are not explicitly defined in the specification, they are useful for distinguishing among several classes of devices:

- The minimal device function is that of a Messaging Server, which is used for Explicit Messaging applications only and acts as a target for Connected and Unconnected Explicit Messages, e.g., for program upload/download, data collection, status monitoring, etc.;
- The next device class is an I/O Server, which adds I/O Messaging Support to a Messaging Server device and acts as a target for both Explicit and I/O Messages, e.g., simple I/O Devices, Pneumatic Valves, AC Drives. These devices are also called Adapters;
- Another device class is a Messaging Client, which adds client support to Messaging Server applications and acts as a target and as an originator for messaging applications, e.g., computer interface cards, HMI devices;
- The most powerful type of device is an I/O Scanner, which adds I/O Message origination support to the functionality of all the other device classes, and which acts as a target and as an originator for Explicit and I/O Messages, e.g., PLCs, I/O Scanners.

3.3.11. Device Profiles

EtherNet/IP devices may utilize any of the profiles described in the CIP Networks Library. As of the publication date of this book, no EtherNet/IP-specific profiles have been defined.

3.3.12. Configuration

EtherNet/IP devices typically come with Electronic Data Sheets (EDS) as described in *Section 2.7*. For EDS-based configuration tools, the EDS should contain a Connection Manager section to describe the details of the connections that can be made into the device. This section basically mirrors what is contained in the Forward_Open message that a Connection Originator would send to the device. Multiple connections can be specified within an EDS that can then be chosen by the configuration tool.

An EDS also may contain individual parameters and/or a Configuration Assembly with a complete description of all parameters within this Assembly. In many applications, the Configuration Assembly is transmitted as an attachment to the Forward_Open message.

3.3.13. Conformance Test

Conformance testing is mandatory for all EtherNet/IP devices. Currently, this test is a protocol conformance test only since it is expected that most implementations use commercially available components for media access and physical attachments.

See Section 6. for more information on Conformance Testing.

3.3.14. Requirements for TCP/IP Support

In addition to the various requirements set forth in the EtherNet/IP Specification, all EtherNet/IP hosts are required to have a minimally functional TCP/IP protocol suite and transport mechanism. The minimum host requirements for EtherNet/IP hosts are those covered in RFC 1122[34], RFC 1123[35], and RFC 1127[36] and the subsequent documents that may supersede them. Whenever a feature or protocol is implemented by an EtherNet/IP host, that feature shall be implemented in accordance to the appropriate RFC (Request for Comment) documents, regardless of whether the feature or protocol is considered required or optional by this specification. The Internet and the RFCs are dynamic. There will be changes to the RFCs and to the requirements included in this section as the Internet and this specification evolve, and these changes will not always provide for backward compatibility.

All EtherNet/IP devices shall at a minimum support:

- Internet Protocol (IP version 4) (RFC 791[27]);
- User Datagram Protocol (UDP) (RFC 768[26]);
- Transmission Control Protocol (TCP) (RFC 793[29]);
- Address Resolution Protocol (ARP) (RFC 826[30]);
- Internet Control Messaging Protocol (ICMP) (RFC 792[28]);
- Internet Group Management Protocol (IGMP) (RFC 1112[33] & 2236[39]);
- IEEE 802.3 (Ethernet) as defined in RFC 894[31].

Although the encapsulation protocol is suitable for use on other networks besides Ethernet that support TCP/IP and products may be implemented on these other networks, conformance testing of EtherNet/IP products is limited to those products on Ethernet. Other suitable networks include:

- Point to Point Protocol (PPP) (RFC 1171[37]);
- ARCNET (RFC 1201[38]);
- FDDI (RFC 1103[32]).

3.3.15. Coexistence of EtherNet/IP and other Ethernet-Based Protocols

EtherNet/IP devices are encouraged, but not required, to support other Ethernet-based protocols and applications not specified in the EtherNet/IP Specification. For example, they may support HTTP, Telnet, FTP, etc. The EtherNet/IP protocol makes no requirements with regard to these protocols and applications.

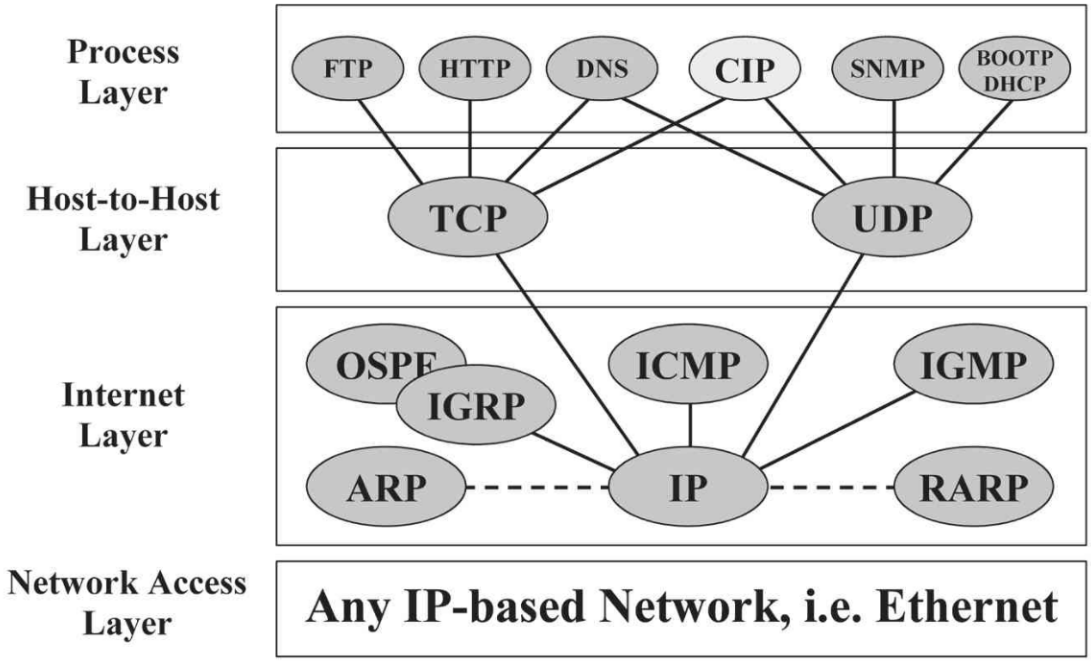


Figure 40 Relationship of CIP to Other Typical Ethernet Protocols

Figure 40 shows the relationship between CIP and other typical Ethernet-based protocol stacks. Since EtherNet/IP, like many other popular protocols, is based on TCP/IP and UDP/IP, coexistence with many other services and protocols is not a problem, and CIP blends nicely into the set of already existing functions. This means that anyone who is already using some or all of these popular Ethernet services can add CIP without undue burden; the existing services like HTTP or FTP may remain as before, and CIP will become another service on the process layer.

3.3.16. Ethernet Infrastructure

To apply EtherNet/IP successfully to the automation world, the issue of determinism has to be considered. The inherent principle of the Ethernet bus access mechanism, whereby collisions are detected and nodes back off and try again later, cannot guarantee determinism. While Ethernet in its present form cannot be made strictly deterministic, there are ways to improve this situation.

First, the hubs typically used in many office environments must be replaced by more intelligent switches that will forward only those Ethernet frames intended for nodes connected to these switches. By using wire-speed switching fabric and full duplex switch technology, collisions are completely avoided; instead of colliding, multiple messages sent to the same node at the same time are queued up inside the switch and are then delivered one after another.

As already mentioned in *Section 3.3.8.2.*, using switches that support IGMP Snooping is highly recommended.

If EtherNet/IP Networks are to be connected to a general company network, this should always be done through a router. The router keeps the UDP multicast messages from propagating into the company network and ensures that the broadcast or multicast office traffic does not congest the control network. Even though the router separates the two worlds, it can be set up to allow the TCP/IP-based Explicit Messages to pass through so that a configuration tool sitting in a PC in the office environment may be capable of monitoring and configuring devices on the control network.

3.3.17. Tools

Tools for EtherNet/IP Networks can be divided into four groups:

- **Physical layer tools** are tools (hardware and/or software) that verify the integrity and conformance of the physical layer or monitor the quality of the data transmission;
- **Commissioning tools**: all EtherNet/IP devices need an IP address. In some cases, setting this address can be obtained only through the Ethernet network *(see Section 3.3.6.)*. In these cases, a BOOTP/DHCP server tool, such as the free BOOTP/DHCP routine downloadable from the Rockwell Automation website, is required.
- **Configuration tools** are software tools capable of communicating with individual devices for data monitoring and configuration purposes. Most configuration tools are EDS-based; however, more complex devices like scanners tend to have their own configuration applets that are only partially based on EDSs. Some of these tools support multiple access paths to the network, e.g., via suitable routing devices. High-level tools also actively query the devices on the network to identify them and monitor their health. Configuration tools also may be integrated into other packages like PLC programming software.
- **Monitoring tools** typically are PC-based software packages (e.g., traffic analyzers or "sniffers") that can capture and display the Ethernet frames on the network. A raw Ethernet frame display may be good enough in some instances, but using a tool that can display both raw Ethernet frames and provide multiple levels of frame interpretation (IP, TCP/UDP, EtherNet/IP header interpretation) is recommended. Due to the popularity of Ethernet, a large number of these tools are available, but not all of them support EtherNet/IP decoding.

In a typical installation, only a commissioning tool and a configuration tool are needed. Protocol monitoring tools are used mainly to investigate interoperability problems and to assist during the development process.

Turn to the EtherNet/IP product catalog on the ODVA website to access a list of vendors that provide tools for EtherNet/IP.

3.3.18. Advice for Developers

Before any development of an EtherNet/IP product is started, the following issues should be considered in detail:

- What functionality (Device Classes, *see Section 3.3.10.*) does the product require today and in future applications?
  - o Messaging server only
  - o Adapter functionality
  - o Messaging client
  - o Scanner functionality
- What are the physical layer requirements? Is IP 65/67 required or is IP 20 good enough?
- Will the development be based on commercially available hardware components and software packages (recommended) or designed from scratch (possible but costly)?
- What are the configuration requirements?
- What design and verification tools should be used?
- What is an absolute must before products can be placed on the market (own the specification, have a Vendor ID, have the product conformance tested)?

Ethernet chipsets and associated base software packages are available from many vendors. For support of the EtherNet/IP part of development, refer to the ODVA website for a list of companies that can support EtherNet/IP development.

3.3.19. EtherNet/IP Summary

Since its introduction in 2000, EtherNet/IP has shown remarkable growth in many applications that previously used traditional networks. This success is largely attributed to the fact that EtherNet/IP, a TCP/UDP/IP-based Ethernet system, introduces real-time behavior into the Ethernet domain without sacrificing any of Ethernet's most useful features, such as company-wide access with standard and specialized tools through corporate networks.

A major strength of EtherNet/IP is the fact that it does not require a modified or highly segregated network: standard switches and routers used in the office world can be used for industrial applications without modification. At the same time, all existing transport-level or TCP/UDP/IP-level protocols can continue to be used without any need for special bridging devices. The substantially improved real-time behavior of CIP Sync and the introduction of CIP Safety also allow EtherNet/IP to be used in applications that currently require several dedicated networks.

Finally, as a member of the CIP family of networks, EtherNet/IP Networks can be combined into an overall CIP Network structure that allows seamless communication among CIP Networks, just as if they were only one network.

4. Benefits of the CIP Family

CIP offers distinct benefits for two groups:

- Device manufacturers
- Users of devices and systems

4.1. Benefits for Device Manufacturers

For device manufacturers, a major benefit of using CIP is the fact that existing knowledge can be re-used from one protocol to another, resulting in lower training costs for development, sales and support personnel. Manufacturers also can reduce development costs, since certain parts (e.g., parameters, profiles) of the embedded firmware that are the same regardless of the network can be re-used from one network to the other. As long as these parts are written in a high-level language, the adaptation is simply a matter of running the right compiler for the new system.

Another important advantage for manufacturers is the easy routing of messages from one system to another. Any routing device can be designed very easily since there is no need to invent a "translation" from one system to another; both systems already speak the same language.

Manufacturers also benefit from working with the same organizations for support and conformance testing.

4.2. Benefits for the Users of Devices and Systems

For users of devices and systems, a major benefit of using CIP is the fact that existing knowledge can be re-used from one protocol to another, e.g. Device Profiles, and device behavior is identical from one system to another, resulting in lower training costs. Technical personnel and users do not have to make great changes to adapt an application from one type of CIP Network to another, and the system integrator can choose the CIP Network that is best suited to his application without having to sacrifice functionality.

Another important CIP advantage is the ease of bridging and routing within The CIP Family. Moving information among incompatible networks is always difficult and cumbersome since it is almost impossible to translate functionality from one network to another. This is where users can reap the full benefits of CIP. Forwarding data and messages from top to bottom and back again is very easy to implement and uses very little system overhead. There is no need to translate from one data structure to another – they are the same! Services and status codes share the same benefit, as these, too, are identical over all CIP Networks. Finally, creating a message that runs through multiple hops of CIP Networks is simply a matter of inserting the full path from the originating to the target device. Not a single line of code or any other configuration is required in the routing devices, resulting in fast and efficient services that are easy to create and maintain. Even though these networks may be used in different parts of the application, messaging from beginning to end really functions as if there is only one network.

Finally, the Producer/Consumer mechanisms used in all CIP Networks provide highly efficient use of transmission bandwidth, resulting in system performance that often is much higher than that of other networks running at higher raw baud rates. With CIP, only the truly important data is transmitted, rather than old data being repeated over and over again.

Planned and future protocol extension will always be integrated in a manner that allows coexistence of “normal” devices with “enhanced” devices like those supporting CIP Sync and/or CIP Safety. Therefore, no strict segmentation into “Standard”, CIP Sync and CIP Safety networks is required unless there is a compelling reason, e.g., unacceptably high response time due to high bus load.

5. Application Layer Enhancements

5.1. CIP Sync and CIP Motion

5.1.1. General Considerations

While CIP Networks[1-4] provide real-time behavior that is appropriate for many applications, a growing number of applications require even tighter control of certain real-time parameters. Let us have a look at some of these parameters:

- Real-Time: This term is being used with a variety of meanings in various contexts. For further use in this section the following definition is used: A system exhibits real-time behavior when it can react to an external stimulus within a predetermined time. How short or how long this time is depends on the application. Demanding industrial control applications require reactions in the millisecond range while, in some process control applications, a reaction time of several seconds or more is sufficient.

- Determinism: A deterministic system allows for a worst-case scenario (not a prediction or a probability) when deciding on the timing of a specific action. Industrial communication systems may offer determinism to a greater or lesser degree, depending on how they are implemented and used. Networks featuring message transmission at a predetermined point in time, such as ControlNet, SERCOS interface and Interbus-S, are often said to offer absolute determinism. On the other hand, networks such as Ethernet may become non-deterministic under certain load conditions; specifically, when it is deployed in half-duplex mode with hubs. However, when Ethernet is deployed with full-duplex, high-speed switches, it operates in a highly deterministic manner *(see 3.3.16)*.
- Reaction Time: In an industrial control system, the overall system reaction time is what determines real-time behavior. The communication system is only one of several factors contributing to the overall reaction time. In general, reaction time is the time from an input stimulus to a related output action.
- Jitter: This term defines the time deviation of a certain event from its average occurrence. Some communication systems rely on very little message jitter, while most applications require only that a certain jitter is not exceeded for actions at the borders of the system, such as input sampling jitter and output action jitter.
- Synchronicity: Distributed systems often require certain actions to occur in a coordinated fashion, i.e., the actions must take place at a predetermined moment in time, independent of where the action is to take place. A typical application is coordinated motion or electronic gearing. Some of these applications require synchronicity in the microsecond range.
- Data Throughput: This is a system's capability to process a certain amount of data within a specific time span. In communication systems, protocol efficiency, the communication model (e.g., Producer/Consumer) and end point processing power are most important, while the wire speed only sets the limit of how much raw data can be transmitted across the physical media.

CIP Sync is a communication principle that enables synchronous low jitter system reactions without the need for low jitter data transmission. This is of great importance in systems that do not provide absolute deterministic data transmission or where it is desirable for a variety of higher layer protocols to run in parallel with the application system protocol. The latter situation is characteristic of Ethernet. Most users of TCP/IP-based Ethernet want to keep using it as before without the need to resort to a highly segregated network segment to run the real-time protocol. The CIP Sync communication principle meets these requirements.

5.1.2. Using IEEE 1588 Clock Synchronization

The published IEEE standard 1588 – Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems[22] – lays the foundation for a precise synchronization of real-time clocks in a distributed system.

An IEEE 1588 system consists of a Time Master that distributes its system time to Time Slaves in a tree-like structure. The Time Master may be synchronized with another real-time clock further up in the hierarchy while the Time Slaves may be Time Masters for other devices "below" them. A Time Slave that is Time Master to another set of devices (typically, in another part of the system) is also called a Boundary Clock. The time distribution is done by multicasting a message with the actual time of the master clock. This message originates in a relatively high layer of the communication stack and, therefore, the actual transmission takes place at a slightly later point in time. Also, the stack processing time varies from one

message to another. To compensate for this delay and its jitter, the actual transmission time can be captured in a lower layer of the communication stack, such as noting the "transmit complete" feedback from the communication chip. This update time capture is then distributed in a follow-up message. The average transmission delay also is determined so that the time offset between the master and slave clock can be compensated.

This protocol has been fully defined for Ethernet UDP/IP systems, and the protocol details for further industrial communication systems will follow. The clock synchronization accuracy that can be achieved with this system depends largely on the precision time capture of the master clock broadcast message. Hardware-assisted time capture systems can reach a synchronization accuracy of 250 ns or less. Some Ethernet chip manufacturers offer integrated IEEE 1588 hardware support.

5.1.3. Additional Object

CIP Sync requires the addition of a new time synchronization object. This object manages the real-time clock inside a CIP Sync device and provides access to the IEEE 1588 timing information. *Figure 41* shows the relationship of the additional object required for CIP Sync.

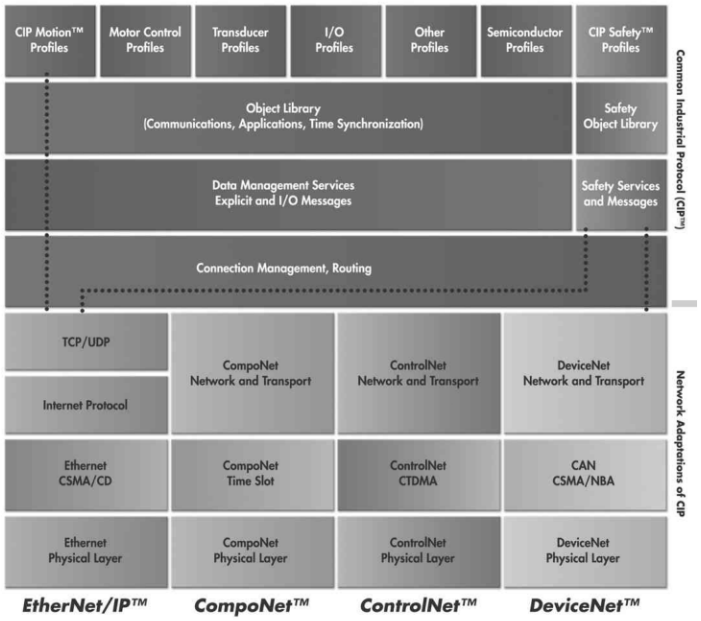


Figure 41 CIP Extensions Required for CIP Sync

5.1.4. CIP Sync Communication Principles

Real-time clocks coordinated through the IEEE 1588 protocol do not, of their own accord, constitute a real-time system yet. Additional details showing how time stamping is used for input sampling and for the coordination of output actions will be added. Some Device Profiles will be extended as well to incorporate time information in their I/O Assemblies. Details of this activity are under discussion in the ODVA Distributed Motion Control JSIG (Joint Special Interest Group).

5.1.5. Message Prioritization

Combining these three elements *(Sections 5.1.2., 5.1.3. and 5.1.4.)* in conjunction with a collision-free infrastructure *(see Section 3.3.16.)* is sufficient to build a real-time system. However, it is necessary to consider all traffic within the system and to arrange all application messages containing time-critical data in such a way that they are guaranteed to arrive at all consumers in time. When other Ethernet protocols—such as HTTP or FTP, which may have very long frames—need to coexist in the same system, careful configuration may be required. Ethernet frames with up to 1,500 bytes of payload (approximately 122 µs long in a 100-Mbit/second system) can easily congest the system and delay important messages by an undetermined amount of time, possibly too long for the system to function correctly.

This is where message prioritization becomes an important element. Of the many prioritization schemes in use or proposed for Ethernet today, EtherNet/IP uses message prioritization according to IEEE 802.3-2002[12]. This is a scheme supported by many switches available today. This message prioritization allows preferential treatment of Ethernet frames in such a way that those frames with the highest priority will jump the message queues in a switch and will be transmitted first. Messages with high priority will be transmitted while those with lower priority typically have to wait. Suitable priority assignments for all time-critical messages then guarantee their preferential treatment. Standard EtherNet/IP and other Ethernet messages will receive low or no priority and thus have to wait until all higher-priority messages have passed. Once this prioritization scheme is implemented, one full-length frame can be accommodated within each communication cycle consisting of a set of prioritized input (port A through port E) and output (port F) messages. *Figure 42* illustrates this process.

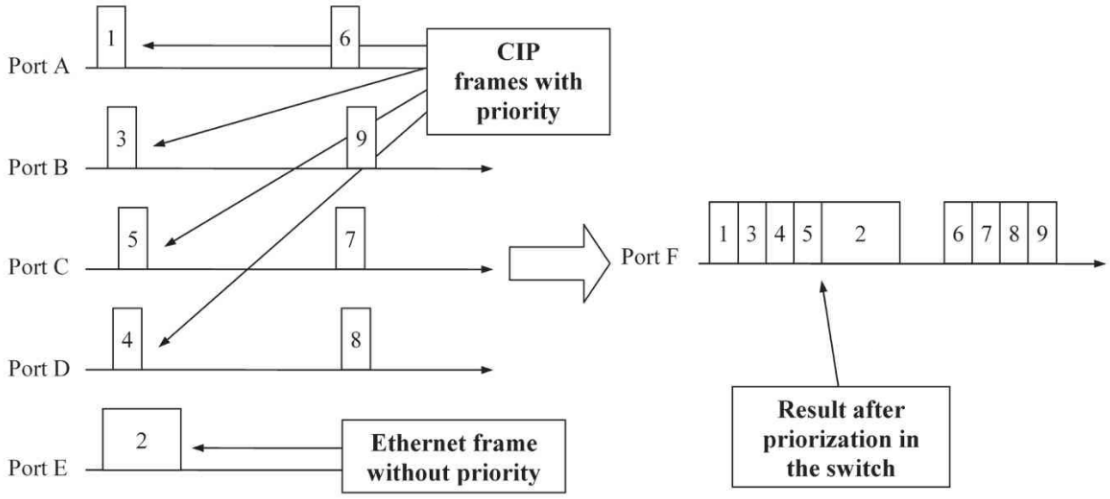


Figure 42 Ethernet Frame Prioritization

5.1.6. Applications of CIP Sync

Typical applications for CIP Sync are time-stamping sensor inputs, distributed time-triggered outputs and distributed motion, such as electronic gearing or camming applications. For example, in motion applications, sensors sample their actual positions at a predetermined time, i.e., in a highly synchronized way, and transmit them to the application master that coordinates the motion. The application master then calculates the new reference values and sends them to the motion drives. Using CIP Sync, the communication system is not required to have

extremely low jitter; it is sufficient to transmit all time-critical messages, and their exact arrival time becomes irrelevant. The assignment of suitable priorities to CIP Sync communication guarantees that all time-critical messages always have the bandwidth they need, and all other traffic automatically is limited to the remaining bandwidth.

As a result of these measures, CIP Sync devices can coexist side by side with other EtherNet/IP devices without any need for network segmentation or special hardware. Even non-EtherNet/IP devices – provided they do not override any of the CIP Sync prioritizations – can be connected without any loss of performance in the CIP Sync application.

5.1.7. Expected Performance of CIP Sync Systems

As mentioned, CIP Sync systems can be built to maintain a synchronization accuracy of better than 250 ns, in many cases without the use of Boundary Clocks. The communication cycle and thus the reaction delay to unexpected events is largely governed by the number of CIP Sync devices in a system. Allowing some bandwidth (approx. 40 %) for non-CIP Sync messages as described in *Section 5.1.5.*, the theoretical limit (close to 100% wire load) for the communication cycle of a CIP Sync system based on a 100-Mbit/s Ethernet network is around 500 µs for 30 coordinated motion axes, with 32 bytes of data each.

5.1.8. CIP Sync Summary

CIP Sync is a natural extension of the EtherNet/IP system into the real-time domain. Unlike many other proposed or existing real-time extensions to other protocols, CIP Sync does not require any strict network segmentation between high-performance, real-time sections and other parts of the communication system. CIP Sync provides the ability to mix parallel TCP/IP based protocols with industrial communication architectures of any size without compromising performance.

CIP Sync currently can be applied to EtherNet/IP, and an extension to the other CIP implementations will follow.

5.2. CIP Safety

Like other safety protocols based on industry standard networks, CIP Safety adds additional services to transport data with high integrity. Unlike other networks, CIP Safety presents a scalable, network-independent approach to safety network design, one in which the safety services are described in a well-defined layer. Since safety functionality is incorporated into each device—rather than in the network infrastructure—CIP Safety allows both standard and safety devices to operate on the same open network. This capability gives users a choice of network architectures—with or without a safety PLC—for their functional safety networks. This approach also enables safety devices from multiple vendors to communicate seamlessly across standard CIP Networks to other safety devices without requiring difficult-to-manage gateways.

A complete definition of all details of CIP Safety can be found in Volume 5[5].

5.2.1. General Considerations

Hardwired safety systems employ safety relays that are interconnected to provide a safety function. Hardwired systems are difficult to develop and maintain for all but the most basic applications. Furthermore, these systems place significant restrictions on the distance between devices.

Because of these issues, as well as distance and cost considerations, implementing safety services on standard communication networks is highly preferable. The key to developing safety networks is not to create a network that cannot fail, but to create a system where failures in the network cause safety devices to go to a known state. If the user knows which state the system will go to in the event of a failure, they can make their application safe. But this means that significantly more checking and redundant coding information is required.

So, to determine the additional safety requirements, the German Safety Bus committee[19] implemented and later extended an existing railway standard[20]. This committee provided design guidelines to safety network developers to allow their networks and safety devices to be certified according to IEC 61508[14].

Based on these standards, CIP was extended for high-integrity safety services. The result is a scalable, routable, network-independent safety layer that alleviates the need for dedicated safety gateways. Since all safety devices execute the same protocol, independent of the media on which they reside, the user approach is consistent and independent of media or network used.

CIP Safety is an extension to standard CIP that has been approved by TÜV Rheinland for use in IEC 61508 SIL 3 and EN 954-1 Category 4 applications. It extends the model by adding CIP Safety application layer functionality, as shown in *Figure 43*. The additions include several safety-related objects and Safety Device Profiles with specific implementation details of CIP Safety as implemented on DeviceNet, otherwise known as DeviceNet Safety.

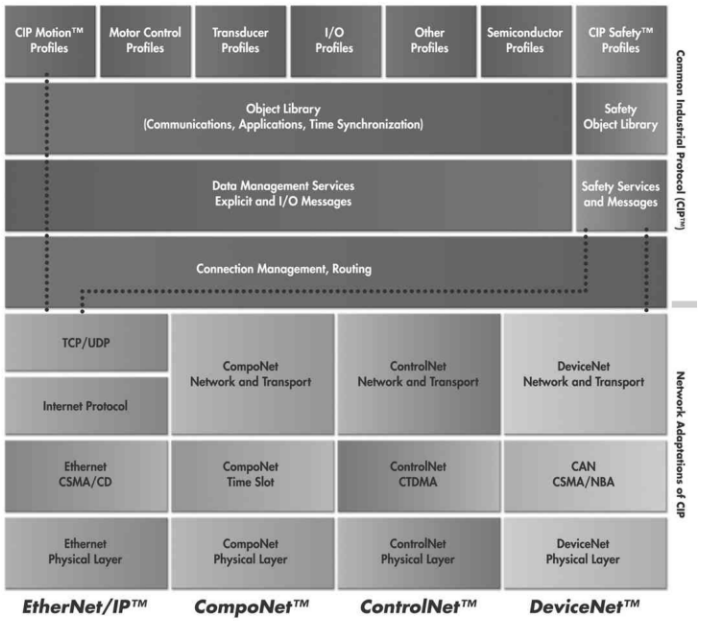


Figure 43 CIP Communication Layers Including Safety

Because the safety application layer extensions do not rely on the integrity *(see Section 5.2.3.)* of the underlying standard CIP as described in *Section 2.* and data link layers as described in *Sections 3.1., 3.2. and 3.3.*, single channel (non-redundant) hardware can be used for the data link communication interface. This same partitioning of functionality allows the use of standard routers for safety data, as shown in *Figure 44*. Routing safety messages is possible because the end device is responsible for ensuring the integrity of the data. If an error occurs during data transmission or in the intermediate router, the end device will detect the failure and take appropriate action.

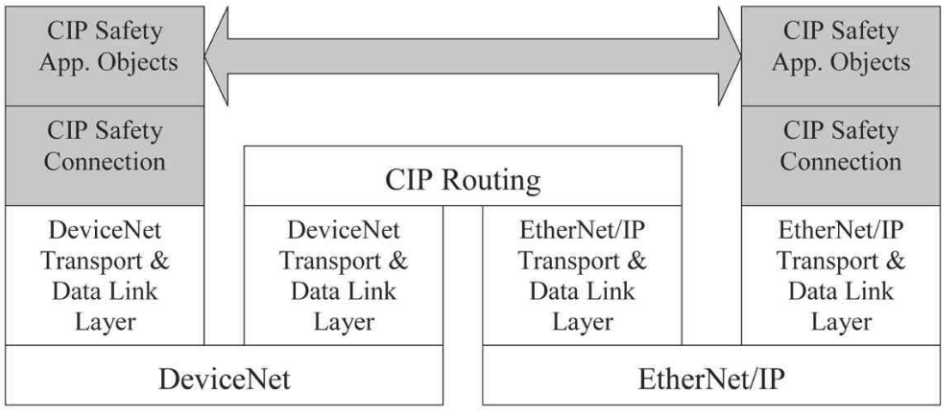


Figure 44 Routing of Safety Data

This routing capability allows the creation of DeviceNet Safety cells with quick reaction times to be interconnected with other cells via a backbone network such as EtherNet/IP for interlocking, as shown in *Figure 45*. Only the safety data that is needed is routed to the required cell, which reduces individual bandwidth requirements. The combination of rapidly responding local safety cells and the inter-cell routing of safety data allows users to create large safety applications with fast response times. Another benefit of this configuration is the ability to multicast safety messages across multiple networks.

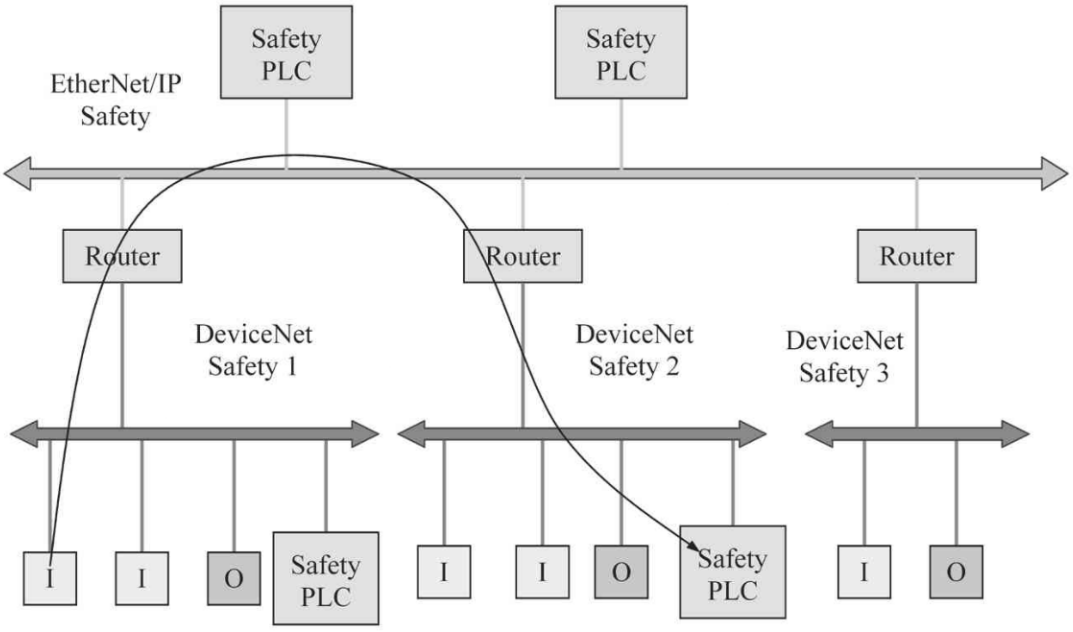


Figure 45 Network Routing

5.2.2. Implementation of Safety

As indicated in *Figure 43*, all CIP Safety devices also have underlying standard CIP functionality. The extension to the CIP Safety application layer is specified using a Safety Validator Object. This object is responsible for managing the CIP Safety Connections (standard CIP Connections are managed through communication objects) and serves as the interface between the safety application objects and the link layer connections, as shown in *Figure 46.* The Safety Validator ensures the integrity of the safety data transfers by applying the measures described in *Section 5.2.3*.

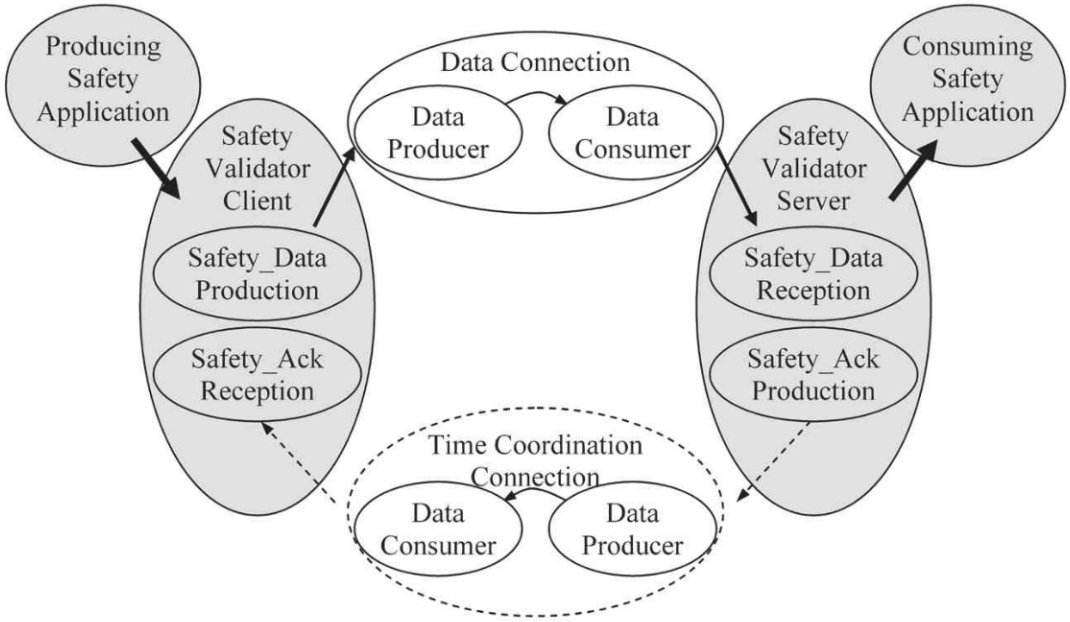


Figure 46 Relationship of Safety Validators

Functions performed by the Safety Validator Object:

- The producing safety application uses an instance of a Client Validator to produce safety data and ensure time coordination;
- The client uses a link data producer to transmit the data and a link consumer to receive time coordination messages;
- The consuming safety application uses a Server Validator to receive and check data;
- The server uses a link consumer to receive data and a link producer to transmit time coordination messages.

The link producers and consumers have no knowledge of the safety packet and fulfill no safety function. The responsibility for high-integrity transfer and checking of safety data lies within the Safety Validators.

5.2.3. Ensuring Integrity

CIP Safety does not prevent communication errors from occurring; rather, it ensures transmission integrity by detecting errors and allowing devices to take appropriate actions. The Safety Validator is responsible for detecting these communication errors. The nine communication errors which must be detected are shown in *Figure 47*, along with the five measures CIP Safety uses to detect these errors[20].

	Measures to detect communication errors				
Communication Errors	Time Expectation via time stamp	ID for send and receive	Safety CRC	Redundancy with Cross Checking	Diverse measure
Message Repetition	X		X*		
Message Loss	X		X*		
Message Insertion	X	X	X*		
Incorrect Sequence	X		X*		
Message Corrupt			X	X	
Message Delay	X				
Coupling of safety and safety data		X			
Coupling of safety and standard data	X	X	X	X	X
Increased age of data in bridge	X				

* The Safety CRC provides additional protection for communication errors in fragmented messages.

Figure 47 Error Detection Measures

5.2.3.1. Time Expectation via Time Stamp

All CIP Safety data are produced with a time stamp that allows Safety Consumers to determine the age of the produced data. This detection measure is superior to the more conventional reception timers. Reception timers can tell how much time has elapsed since a message was last received, but they do not convey any information about the actual age of the data. A time stamp allows transmission, media access/arbitration, queuing, retry and routing delays to be detected.

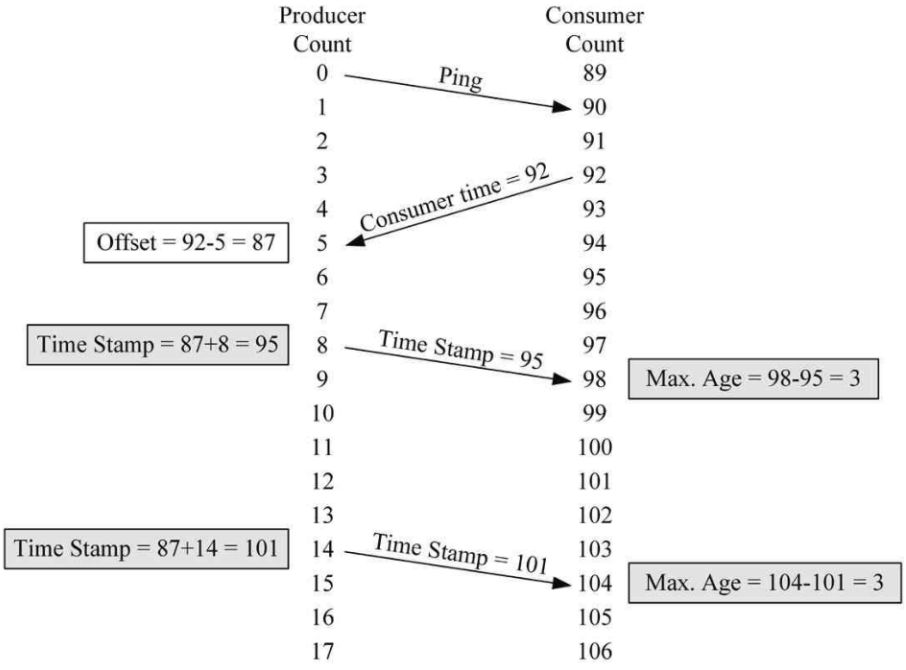


Figure 48 Time Stamp

Time is coordinated between producers and consumers using ping requests and ping responses, as shown in *Figure 48*. After a connection is established, the producer generates a ping request, which causes the consumer to respond with its consumer time. The producer will note the time difference between the ping production and the ping response and store this as an offset value. The producer will add this offset value to its producer time for all subsequent data transmissions. This value is transmitted as the time stamp. When the consumer receives a data message, it subtracts its internal clock from the time stamp to determine the data age. If the data age is less than the maximum age allowed, the data are applied; otherwise, the connection goes to the safety state. The device application is notified so that the connection safety state can be reflected accordingly.

The ping request-and-response sequence is repeated periodically to correct for any producer or consumer time base drift.

5.2.3.2. Production Identifier (PID)

A Production Identifier is encoded in all data produced by a Safety Connection to ensure that each received message arrives at the correct consumer. The PID is derived from an electronic key, the device Serial Number and the CIP Connection Serial Number. Any safety device inadvertently receiving a message with the incorrect PID will go to a safety state. Any safety device that does not receive a message within the expected time interval with the correct PID will also go to a safety state. This measure ensures that messages are routed correctly in multi-network applications.

5.2.3.3. Safety CRC (Cyclic Redundancy Code)

All safety transfers on CIP Safety use Safety CRCs to ensure the integrity of the transfer of information. The Safety CRCs serve as the primary means of detecting possible corruption of transmitted data. They provide detection up to a Hamming distance of four for each data transfer section, though the overall Hamming distance coverage is greater for the complete transfer due to the protocol's redundancy. The Safety CRCs are generated in the Safety Producers and checked in the Safety Consumers. Intermediate routing devices do not examine the Safety CRCs. Thus, by employing end-to-end Safety CRCs, the individual data link CRCs are not part of the safety function. This eliminates certification requirements for intermediate devices and helps to ensure that the safety protocol is independent of the network technology. The Safety CRC also provides a strong protection mechanism that allows detection of underlying data link errors, such as bit stuffing or fragmentation.

While the individual link CRCs are not relied on for safety, they are still enabled. This provides an additional level of protection and noise immunity by allowing data retransmission for transient errors at the local link.

5.2.3.4. Redundancy and Cross-check

Data and CRC redundancy with cross-checking provides an additional measure of protection by detecting possible corruption of transmitted data. By effectively increasing the Hamming distance of the protocol, these measures allow long safety data packets—up to 250 bytes—to be transmitted with high integrity. For short packets of two bytes or less, data redundancy is not required; however, redundant CRCs are cross-checked to ensure integrity.

5.2.3.5. Diverse Measures for Safety and Standard

The CIP Safety protocol is present only in safety devices, which prevents standard devices from masquerading as safety devices.

5.2.4. Safety Connections

CIP Safety provides two types of Safety Connections:

- Unicast Connections
- Multicast Connections

A Unicast Connection, as shown in *Figure 49*, allows a Safety Validator Client to be connected to a Safety Validator Server using two link layer connections.

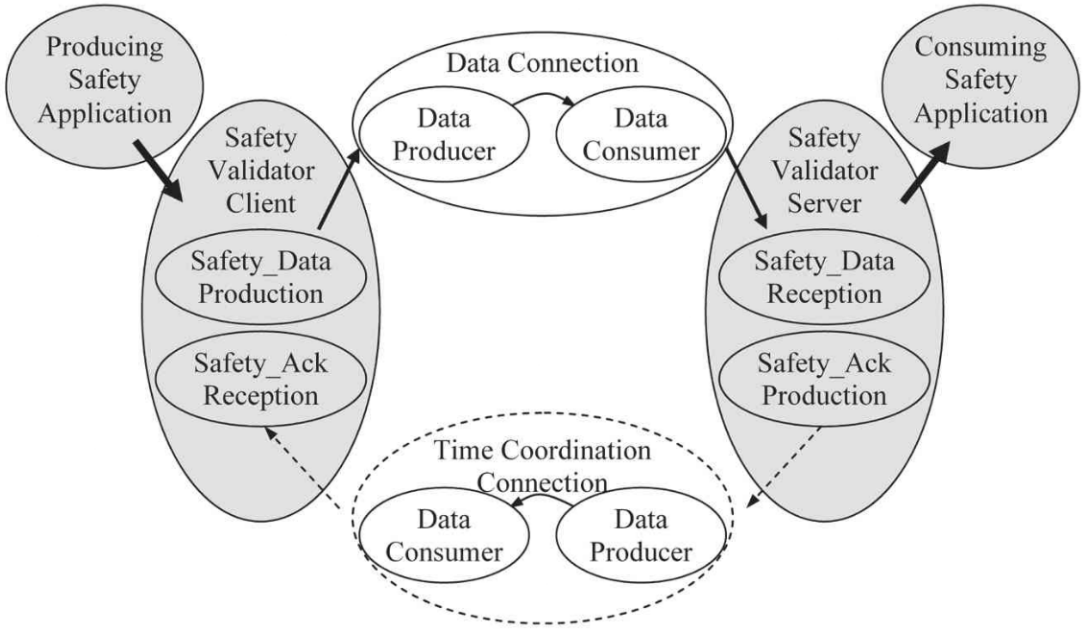


Figure 49 Unicast Connection

A Multicast Connection, as shown in *Figure 50*, allows up to 15 Safety Validator Servers to consume safety data from a Safety Validator Client. When the first Safety Validator Server establishes a connection with a Safety Validator Client, a pair of link layer connections are established, one for data-and-time correction and one for time coordination. Each new Safety Validator Server uses the existing data-and-time correction connection and establish a new time coordination connection with the Safety Validator Client.

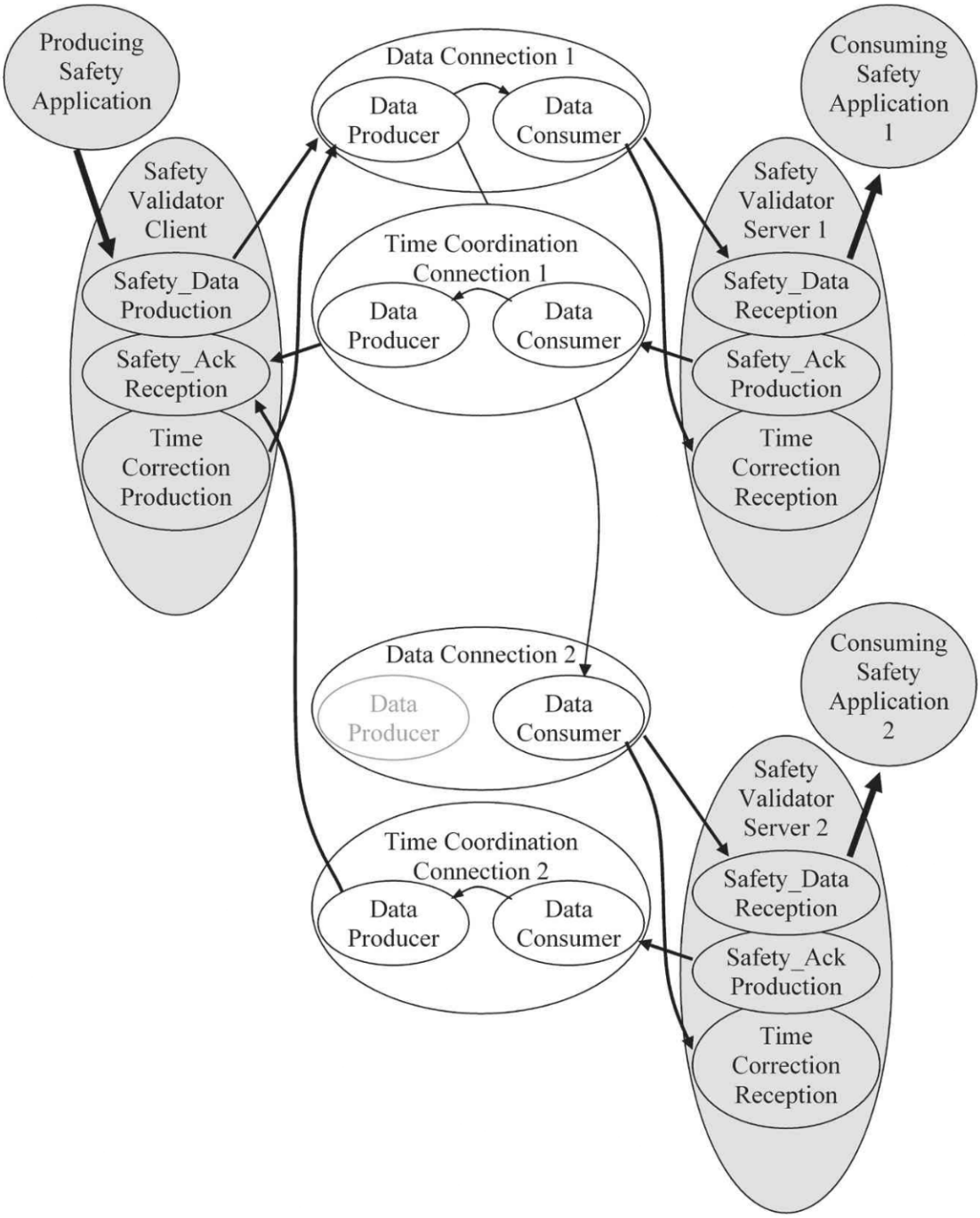


Figure 50 Multicast Connection

To optimize throughput on DeviceNet, each Multicast Connection uses three data link connections, as shown in *Figure 51*. The data-and-time correction messages are sent on separate connections. This allows short messages to be transmitted on DeviceNet within a single CAN frame and reduces the overall bandwidth, since the time correction and time coordination messages are sent at much slower periodic intervals.

When multicast messages are routed off-link, the router combines the data-and-time correction messages from DeviceNet and separates them when messages reach DeviceNet. Since the safety message contents are unchanged, the router provides no safety function.

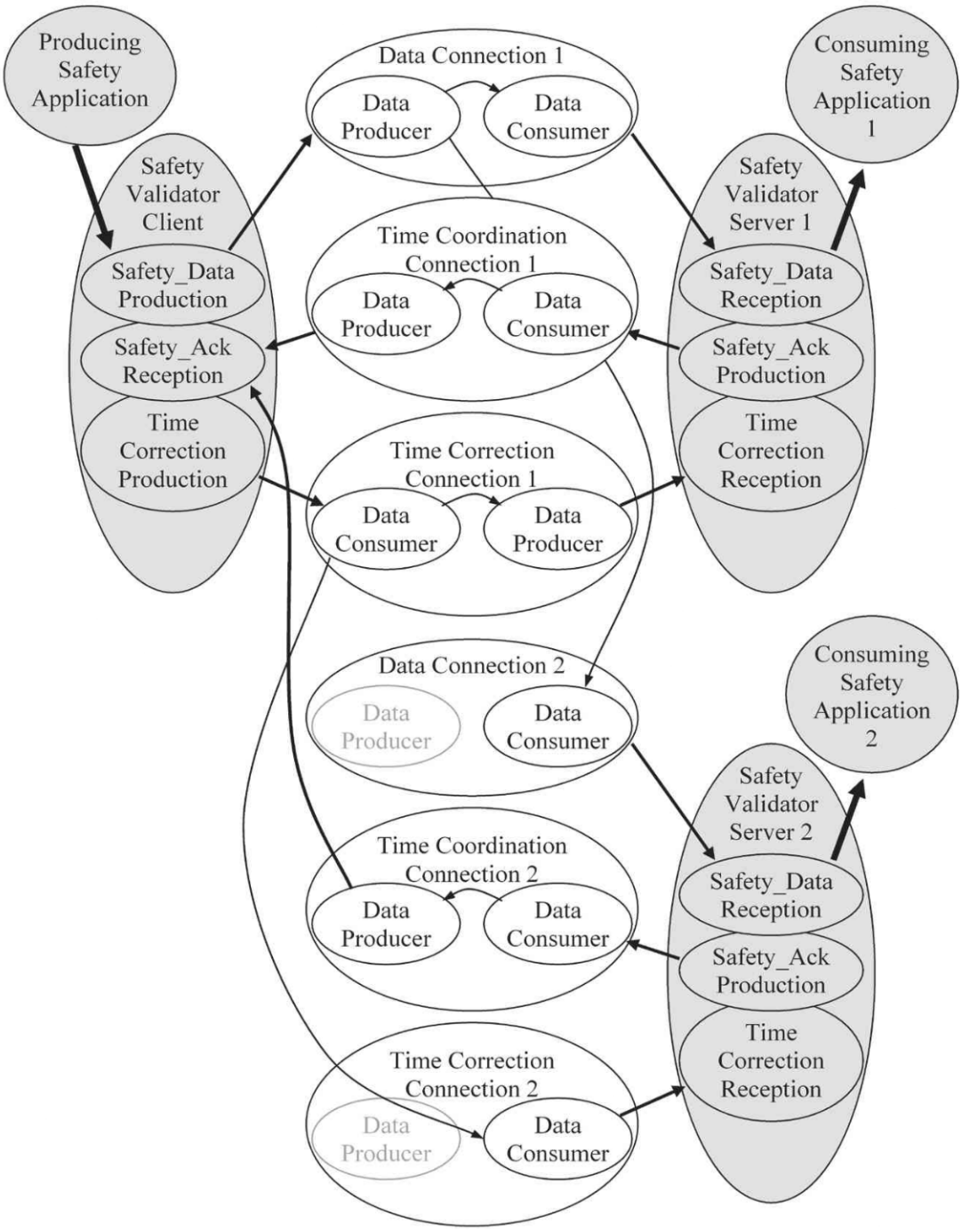


Figure 51 Multicast Connection on DeviceNet

5.2.5. Message Packet Sections

CIP Safety has four message sections:

- Data
- Time-stamp
- Time correction
- Time coordination

The description of these formats goes beyond the scope of this book. Available materials on this topic that go into more detail[5, 21].

5.2.6. Configuration

Before safety devices can be used in a safety system, they first must be configured and connections must be established. The process of configuration requires placement of configuration data from a configuration tool in a safety device. There are two possible sequences for configuration:

- Configuration tool directly to device, or
- Via an intermediate device.

In the configuration tool-to-device case, as shown in *Figure 52*, the configuration tool writes directly to the device to be configured[1-2].

In the case of intermediate device configuration, the tool first writes to an originator[1] and the originator writes to the target using an Originator-to-Target Download[3] or a Safety_Open service[4]. The Safety_Open service[4] is unique in that it allows a safety connection to be established at the same time that a device is configured.

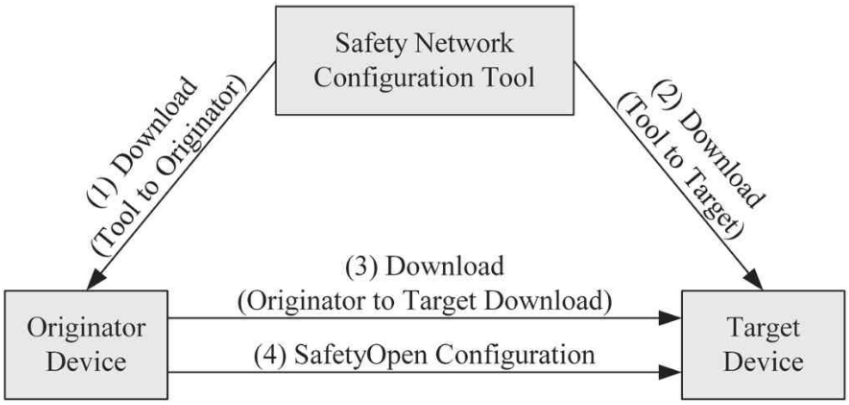


Figure 52 Configuration Transfers

5.2.7. Connection Establishment

CIP provides a connection establishment mechanism, using a Forward_Open service that allows producer-to-consumer connections to be established locally or across multiple networks via intermediate routers. An extension of the Forward_Open, called the Safety_Open service, has been created to allow the same multi-network connections for safety.

There are two types of Safety_Open requests:

- Type 1: With configuration
- Type 2: Without configuration

With the Type 1 Safety_Open request, configuration and connections are established at the same time, allowing rapid configuration of devices with simple and relatively small configuration data.

With the Type 2 Safety_Open request, the safety device first must be configured and the Safety_Open request then establishes a Safety Connection. This separation of configuration and connection establishment allows the configuration of devices with large and complex configuration data.

In both cases, the Safety_Open request establishes all underlying link layer connections – across the local network as well as any intermediate networks and routers.

5.2.8. Configuration Implementation

CIP Safety provides the following protection measures to ensure the iconfiguration integrity:

- Safety Network Number
- Password Protection
- Configuration Ownership
- Configuration Locking

5.2.8.1. Safety Network Number

The Safety Network Number provides a unique network identifier for each network in the safety system. The Safety Network Number, combined with the local device address, allows any device in the safety system to be uniquely addressed.

5.2.8.2. Password Protection

All safety devices support the use of an optional password. The password mechanism provides an additional protection measure, prohibiting the reconfiguration of a device without the correct password.

5.2.8.3. Configuration Ownership

The owner of a CIP Safety device can be specified and enforced. Each safety device can specify that it be configured only by a selected originator, or that the configuration is accomplished by a configuration tool.

5.2.8.4. Configuration Locking

Configuration Locking provides the user with a mechanism to ensure that all devices have been verified and tested prior to being used in a safety application.

5.2.9. Safety Devices

The relationship of the objects within a safety device is shown in *Figure 53*. Note that CIP Safety extends the CIP object model, with the addition of Safety I/O Assemblies, Safety Validator and Safety Supervisor Objects.

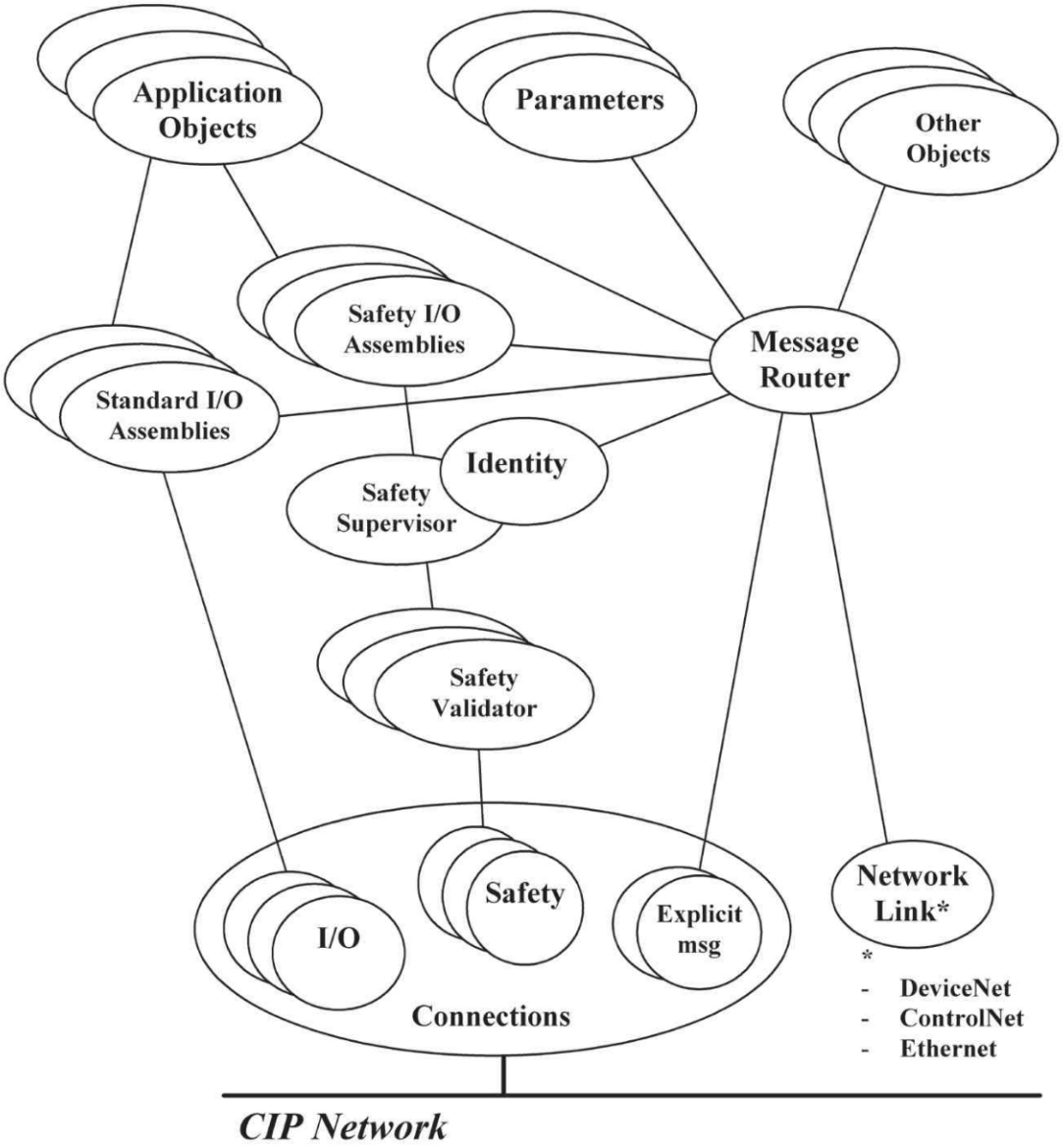


Figure 53 Safety Device Objects

5.2.10. Safety Supervisor

The Safety Supervisor Object provides a common configuration interface for safety devices. The Safety Supervisor Object centralizes and coordinates application object state behavior and related status information, and exception status indications (alarms and warnings), and defines a behavior model that identified objects assume belong to safety devices.

....5.2.11. CIP Safety Summary

CIP Safety is a scalable, routable, network-independent safety protocol based on extensions to the CIP architecture. This concept can be used in solutions ranging from device-level networks such as DeviceNet to higher level networks such as EtherNet/IP. Designing network independence into CIP Safety allows multi-network routing of Safety Connections. Functions such as multi-network routing and multicast messaging provide a strong foundation that enables users to create the rapidly responding local cells and interconnect remote cells that are required for today's safety applications. CIP Safety's design also enables expansion to future network technologies as they become available.

6. Conformance Testing

Manufacturers can have their devices tested for conformance with CIP Networks specifications in one of several independent conformance test centers. Only then do two key characteristics of all CIP Networks implementations become possible: interoperability and interchangeability.

- **Interoperability** means that devices from all manufacturers can be configured to operate with each other on the network.
- **Interchangeability** goes one step further by providing the means for devices of the same type (i.e., they comply with the same Device Profile) to be logical replacements for each other, regardless of the manufacturer.

The **conformance test** checks both of these characteristics. This test is divided into three parts:

- A **software test** verifies the function of the network protocol. Depending on the complexity of the device, up to several thousand messages are transmitted to the device under test (DUT). To ensure a test that is closely adapted to the characteristics of the DUT, the manufacturer must provide a formal description of all relevant features of the DUT.
- A **hardware test** examines conformance with the characteristics of the physical layer. This test checks all requirements of the network specification
- A **system interoperability test** that verifies that the device can function in a well-populated network that has a variety of devices from various manufacturers.

A software test suite is available from ODVA for EtherNet/IP and DeviceNet, and from ControlNet International for ControlNet devices. Each is a Windows®-based tool that uses network interface cards in the PC to access the device under test. It is recommended that device developers run this test in their own lab before taking devices to the official test center. The hardware test (where appropriate) and the system interoperability test involve more complex test setups that typically are not available to device developers. When a device passes the test, it is said to be CONFORMANCE TESTED. A device that has not been tested accordingly has a significant market disadvantage, and may in fact be in violation of the specific network's Terms of Usage Agreement. Devices that have passed conformance testing are published on the ODVA or ControlNet International website.

Figure 54 Certification Marks

7. Endnotes

[1] CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 2.1, January 2005, © 2001 through 2005 by Open DeviceNet Vendor Association.

[2] CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.1, January 2005, © 1999 through 2005 by Open DeviceNet Vendor association.

[3] CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.1, January 2005, © 1994 through 2005 by Open DeviceNet Vendor Association.

[4] CIP Networks Library, Volume 4, ControlNet Adaptation of CIP, Edition 1.0, April 2005, © 1997 through 2005 by ControlNet International.

[5] CIP Networks Library, Volume 5, CIP Safety, Edition 1.0, January 2005, © 2005 ODVA.

[6] Planning and Installation Manual, DeviceNet Cable System, Publication number PUB00027R1, downloadable from ODVA website (http://www.odva.org/)

[7] Recommended IP Addressing Methods for EtherNet/IP Devices, Publication number PUB00028R0, downloadable from ODVA website (http://www.odva.org/)

[8] IEC 61131-3:1993 – Programmable controllers – Part 3: Programming languages.

[9] Controller Area Network – Basics, Protocols, Chips and Application. IXXAT Automation, 2001. ISBN 3-00-007376-0.

[10] ISO 11898:1993 – Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication.

[11] IEEE 802.3:2000, ISO/IEC 8802-3:2000 – IEEE Standard for Information technology – Local and metropolitan area networks – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specification.

[12] IEEE 802.3-2002 – Information Technology – Telecommunication & Information Exchange Between Systems – LAN/MAN – Specific Requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications 2002.

[13] ISO/IEC 7498-1:1994 – ISO/IEC Standard - Information technology – Open Systems Interconnection – Basic Reference Model.

[14] IEC 61508:1998 – Functional safety of electrical/electronic/programmable electronic safety-related systems.

[15] IEC 62026-3:2000 – Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet

[16] EN 50325-2:2000 – Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces – Part 2: DeviceNet

[17] Chinese national standard: GB/T 18858:2003 – Low-voltage Switchgear and Controlgear Controller-Device Interface

[18] IEC 61158:2000 – Digital data communications for measurement and control – Network for use in industrial control systems

[19] Draft proposal test and certification guideline, safety bus systems, BG Fachausschuß Elektrotechnik 28-May-2000

[20] EN 50159-1:2001 – Railway applications, communication, signaling and processing systems.

[21] David A. Vasko, Suresh R. Nair: CIP Safety: Safety Networking for the Future; Proceedings of the 9th International CAN Conference 2003.

[22] IEEE 1588-2002 – Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

[23] Viktor Schiffer: Modular EDSs and other EDS Enhancements for DeviceNet; Proceedings of the 9th International CAN Conference 2003.

[24] Viktor Schiffer: Device Configuration Using Electronic Data Sheets, ODVA 2003 Conference and 9th Annual Meeting, downloadable from ODVA website.

[25] Viktor Schiffer, Ray Romito: DeviceNet Development Considerations, downloadable from ODVA website, 2000.

[26] RFC 768 – User Datagram Protocol, 1980.

[27] RFC 791 – Internet Protocol, 1981.

[28] RFC 792 – Internet Control Message Protocol, 1981.

[29] RFC 793 – Transmission Control Protocol, 1981.

[30] RFC 826 – Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, 1982.

[31] RFC 894 – Standard for the transmission of IP datagrams over Ethernet networks, 1984.

[32] RFC 1103 – Proposed standard for the transmission of IP datagrams over FDDI Networks, 1989.

[33] RFC 1112 – Host extensions for IP multicasting, 1989.

[34] RFC 1122 – Requirements for Internet Hosts - Communication Layers, 1989

[35] RFC 1123 – Requirements for Internet Hosts - Application and Support, 1989.

[36] RFC 1127 – Perspective on the Host Requirements RFCs, 1989.

[37] RFC 1171 – Point-to-Point Protocol for the transmission of multi-protocol datagrams over Point-to-Point links, 1990.

[38] RFC 1201 – Transmitting IP traffic over ARCNET networks, 1991.

[39] RFC 2236 – Internet Group Management Protocol, Version 2, 1997.

All RFCs are downloadable from http://www.faqs.org/rfcs/

8. Bibliography

CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 2.1, January 2005, © 2001 through 2005 by Open DeviceNet Vendor Association.

CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.1, January 2005, © 1999 through 2005 by Open DeviceNet Vendor association.

CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.1, January 2005, © 1994 through 2005 by Open DeviceNet Vendor Association.

CIP Networks Library, Volume 4, ControlNet Adaptation of CIP, Edition 1.0, April 2005, © 1997 through 2005 by ControlNet International.

CIP Networks Library, Volume 5, CIP Safety, Edition 1.0, January 2005, © 2005 ODVA.

9. Related Publications

CIP Networks Library, Volume 1, Common Industrial Protocol, Edition 2.1, January 2005, © 2001 through 2005 by Open DeviceNet Vendor Association.

CIP Networks Library, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.1, January 2005, © 1999 through 2005 by Open DeviceNet Vendor association.

CIP Networks Library, Volume 3, DeviceNet Adaptation of CIP, Edition 1.1, January 2005, © 1994 through 2005 by Open DeviceNet Vendor Association.

CIP Networks Library, Volume 4, ControlNet Adaptation of CIP, Edition 1.0, April 2005, © 1997 through 2005 by ControlNet International.

CIP Networks Library, Volume 5, CIP Safety, Edition 1.0, January 2005, © 2005 ODVA.

Planning and Installation Manual, DeviceNet Cable System, Publication number PUB00027R1, downloadable from ODVA website (http://www.odva.org/)

EtherNet/IP Planning and Installation Manual, Publication number PUB00148R0, downloadable from ODVA website (http://www.odva.org/)

EtherNet/IP Implementor Workshop Documents - Performance Test Terminology for EtherNet/IP Devices, Publication number PUB00080R1.1, downloadable from ODVA website (http://www.odva.org/)

CAN Specifications – Controller Area Network, Version 2.0, 1991, Robert Bosch GmbH. http://www.semiconductors.bosch.de/pdf/can2spec.pdf

10. Abbreviations

For the purposes of this standard, the following abbreviations apply.

Abbreviation	Meaning
ASCII	American Standard Code for Information Interchange
CIP	The Common Industrial Protocol defined by Volume 1 of the CIP Networks Library.
CID	Connection Identifier
EPR	Expected Packet Rate
ISO	International Standards Organization
MAC ID	Media Access Control Identifier (another name for Network Address)
ODVA	Open DeviceNet Vendor Association, Inc.
OSI	Open Systems Interconnection (see ISO 7498)
UCMM	Unconnected Message Manager
CRC	Cyclic Redundancy Check
LED	Light Emitting Diode.
MAC	Media Access Control sublayer
NAP	Network Access Port
NUT	Network Update Time
RG-6	Standard for coaxial cable
SMAX	MAC ID of the maximum scheduled node
UMAX	MAC ID of maximum unscheduled node
FTP	File transfer protocol. An internet application that uses TCP reliable packet transfer to move files between different nodes. (not to be confused with STP/FTP)
RFC	Request For Comments (RFC) – The document series, begun in 1969, which describes the Internet suite of protocols and related experiments. Not all (in fact, very few) RFCs describe the Internet Standards, but all Internet Standards are written up as RFCs. The RFC series of documents is unusual in that the proposed protocols are forwarded by the Internet research and development community, acting on their own behalf, as opposed to the formally reviewed and standardized protocols that are promoted by organizations such as CCITT and ANSI. [Source: RFC 1392]
TCP	Transmission Control Protocol (TCP) - An Internet Standard transport layer protocol defined in STD 7, RFC 793. It is connection-oriented and stream-oriented, as opposed to UDP. See also: connection-oriented, stream-oriented, User Datagram Protocol. [Source: RFC1392]
UDP	User Datagram Protocol (UDP) - An Internet Standard transport layer protocol defined in STD 6, RFC 768. It is a connectionless protocol which adds a level of reliability and multiplexing to IP. See also: connectionless, Transmission Control Protocol. [Source: RFC1392]

11. Definitions

For the purposes of this document, the following definitions apply:

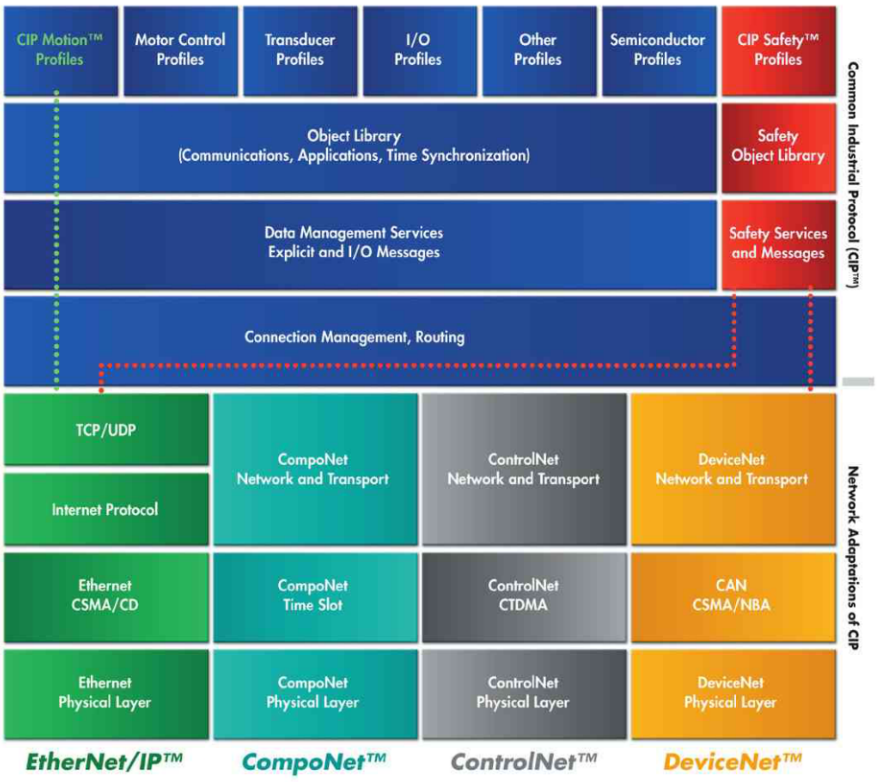
Term	Definition
allocate	In the DeviceNet context, this is the process of reserving resources of the Predefined Master/Slave Connection Set in a DeviceNet node. It is associated with services of a similar name, of the DeviceNet Object Class (Class ID 0x03).
application	Typically refers to the application layer of the ISO-OSI model. The application layer is the part of the product that performs application-specific functions. Typically, application objects that provide the desired behavior are associated with the application.
attribute	A description of an externally visible characteristic or feature of an object. The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes also may affect the behavior of an object. Attributes are divided into class attributes and instance attributes.
behavior	Indication of how the object responds to particular events. Its description includes the relationship between attribute values and services.
bit	A unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted.
broadcast	A message that is sent to all nodes on a network. It also refers to the property of a network where all nodes listen to all messages transmitted for purposes of determining bus access/priority.
byte	A sequence of 8 bits that is treated as a single unit.
class	A set of objects all of which represent a similar system component. A class is a generalization of the object, a template for defining variables and methods. All objects in a class are identical in form and behavior, but they may contain different attribute values.
Object-specific service	A service defined by a particular object class to perform a required function that is not performed by any common services. A class-specific service is unique to the object class that defines it.
client	(1) An object that uses the services of another (server) object to perform a task. (2) An initiator of a message to which a server reacts.
connection	A logical binding between two application objects. These application objects may be the same or different devices.

Term	Definition
Connection Identifier	Identifier assigned to a transmission that is associated with a particular connection between producers and consumers that identifies a specific piece of application information.
connection path	Is made up of a byte stream that defines the application object to which a connection instance applies.
consumer	A node that is receiving (i.e.: consumes) data from a producer.
consuming application	The application that consumes data.
CRC error	Error that occurs when the cyclic redundancy check (CRC) value does not match the value generated by the transmitter.
cyclic	Term used to describe events that repeat in a regular and repetitive manner.
datagram	A transmitted message.
device	A physical hardware connection to the link. A device may contain more than one node.
Device Profile	A collection of device-dependent information and functionality providing consistency between similar devices of the same device type.
drop or drop line	The cable that connects one or more nodes to a trunk cable, usually accomplished using a tap.
encapsulation	The technique used by layered protocols in which a layer adds header information to the protocol data unit (PDU) from the layer above for purposes of carrying one protocol within another.
Ethernet	A standard for LANs, initially developed by Xerox, and later refined by Digital, Intel and Xerox (DIX). In its original form, all hosts are connected to a coaxial cable where they contend for network access using a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) paradigm. See also: 802.x, Local Area Network, token ring. [Source: RFC1392]
Expected Packet Rate	The basic rate at which a connection transmits its data.
fixed tag	A two-byte field in a ControlNet Lpacket that identifies unconnected or station management services the node is expected to perform. The first byte is the specific service code and the second byte contains the MAC ID of the destination node.
frame	See MAC Frame
generic tag	A three-byte field in a ControlNet Lpacket that serves as the Connection Identifier (CID). It is associated with a specific piece of application information.

Term	Definition
Guardband	It is the portion of ControlNet bandwidth that is allocated for the transmission of the moderator frame.
instance	The actual physical presentation of an object within a class. Identifies one of potentially many objects within the same object class.
Keeper	Object responsible for holding and distributing the Connection Originator schedule data for all Connection Originator devices on a ControlNet Network.
Link or Data Link	Refers to the Data Link layer of the ISO/OSI model.
Lpacket	On ControlNet, the Lpacket (or link packet) is a portion of the MAC Frame where application information that contains a size, control byte, tag, and link data is transmitted. There may be one or more Lpackets in a single MAC Frame.
MAC frame	A collection of MAC symbols transmitted on the network medium that contains the required message formatting/framing necessary to pass a message to another node. For example, a ControlNet MAC Frame consists of a preamble, start delimiter, source MAC ID, Lpackets, CRC, and end delimiter.
Message Router	The object within a node that distributes explicit message requests to the appropriate application objects.
multicast	A packet that is sent to multiple nodes on the network.
network	A series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters and bridges.
Network Access Port	On ControlNet, this is an alternate physical layer connection point on a permanent node that allows a temporary node to be connected to the link. The temporary node has its own network address, but simply shares the permanent node's physical layer connection to the network.
network address	An integer identification value assigned to each node on a CIP network.
network status indicators	Indicators (i.e.: LEDs) on a node indicating the status of the Physical and Data Link Layers.
Network Update Time	Repetitive time interval on a ControlNet Network that is used to subdivide the network bandwidth. It determines the fastest rate that real-time data can be transferred on the network.
node	A connection to a link that requires a single MAC ID.

Term	Definition
object	(1) An abstract representation of a particular component within a product. Objects can be composed of any or all of the following components: a) data (information which changes with time); b) configuration (parameters for behavior); c) methods (things that can be done using data and configuration). (2) A collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior.
object-specific service	A service defined by a particular object class to perform a required function that is not performed by one of the common services. An object-specific service is unique to the object class that defines it.
originator	The client responsible for establishing a connection path to the target.
point-to-point	A one-to-one data exchange relationship between two, and only two nodes.
port	A CIP port is the abstraction for a physical network connection to a CIP device. A CIP device has one port for each network connection. Within the EtherNet/IP specific context, a TCP or UDP port is a transport layer de-multiplexing value. Each application has a unique port number associated with it. [Source: RFC1392].
point-to-point	A one-to-one data exchange relationship between two, and only two nodes.
port	A CIP port is the abstraction for a physical network connection to a CIP device. A CIP device has one port for each network connection. Within the EtherNet/IP specific context, a TCP or UDP port is a transport layer de-multiplexing value. Each application has a unique port number associated with it. [Source: RFC1392].
producer	A node that is responsible for transmitting data.
redundant media	A system using more than one medium to help prevent communication failures.
repeater	Two-port active Physical Layer device that reconstructs and retransmits all traffic on one segment to another segment.
scheduled	On ControlNet these are data transfers that occur in a deterministic and repeatable manner, on preconfigured NUT-based intervals.

Term	Definition
segment	This term has two uses within CIP. With respect to cable topology, a segment is a length of cable connected via taps with terminators at each end; a segment has no active components and does not include repeaters. With respect to messaging, segments (Logical Segments, Port Segments, etc.) are used in explicit messages to describe various addressing elements of devices such as: Class IDs, Attribute IDs, ports, Connection Points, etc.
Serial Number	A unique 32-bit integer assigned by each manufacturer to every device. The number need only be unique with respect to the manufacturer.
server	A device or object that provides services to another device (the client).
service	Operation or function that an object performs upon request from another object.
tap	Point of attachment on the trunk where one or more drop lines are attached.
target	The end-node to which a connection is established.
terminator	A resistor placed at the physical extreme ends of trunk segments to prevent transmission reflections from occurring.
transceiver	The physical component within a node that provides transmission and reception of signals onto and off of the medium.
trunk or trunk line	The main bus or central part of a cable system, typically terminated at each end by a termination resistor.
Unconnected Message Manager	The function within a node that transmits and receives Unconnected Explicit Messages.
unscheduled	On ControlNet, this refers to data transfers that use the unscheduled portion of the NUT.


CIP Motion™ Profiles
Motor Control Profiles
Transducer Profiles
I/O Profiles
Other Profiles
Semiconductor Profiles
CIP Safety™ Profiles
Object Library (Communications, Applications, Time Synchronization)
Safety Object Library
Data Management Services Explicit and I/O Messages
Safety Services and Messages
Connection Management, Routing
Common Industrial Protocol (CIP™)
TCP/UDP
Internet Protocol
CompoNet Network and Transport
ControlNet Network and Transport
DeviceNet Network and Transport
Ethernet CSMA/CD
CompoNet Time Slot
ControlNet CTDMA
CAN CSMA/NBA
Ethernet Physical Layer
CompoNet Physical Layer
ControlNet Physical Layer
DeviceNet Physical Layer
Network Adaptations of CIP
EtherNet/IP™
CompoNet™
ControlNet™
DeviceNet™